CLOMOSY
BULUT TABANLI MOBİL UYGULAMA GELİŞTİRME

Atiker Yazılım
"İşletmeye Değer Katan Yatırım"
Ömer Atiker Holding A.Ş.

Atiker Yazılım Veri İşlem A.Ş.
Adres: Başkule Plaza, Mahmutbey, İnönü Cd. No:122D, 34218 Bağcılar/İstanbul
E-posta: contact@clomosy.com
Telefon: 0 (212) 461 00 00

ISBN 978-625-95752-0-9

İÇİNDEKİLER

BÖLÜM 7: DİZİLER

BÖLÜM 8: PROSEDÜR (VOID) FONKSİYONLAR VE HAZIR KÜTÜPHANE FONKSİYONLARI

BÖLÜM 9: HATA YAKALAMAK

BÖLÜM 10: FORMLAR

BÖLÜM 11: BİLEŞENLER

BÖLÜM 12: OLAYLAR (EVENTS)

BÖLÜM 13: BİRİMLER (UNITS)

GÖRSEL TABLOSU

KOD TABLOSU

TABLOLAR

ÖN SÖZ

Dünya, dijitalleşmenin etkisiyle her geçen gün yeniden şekilleniyor. Teknoloji, artık yalnızca iş dünyasının değil, bireylerin, kurumların ve toplumların geleceğe dair vizyonlarını belirleyen bir güç haline geldi. Küresel rekabetin giderek arttığı bu dönemde, değişime uyum sağlamak ve yenilikçi yaklaşımlar geliştirmek yalnızca bir tercih değil, sürdürülebilirliğin temel şartıdır.
OMA Holding olarak, bu değişim sürecinde teknolojiyle şekillenen geleceğin mimarları arasında yer almayı bir sorumluluk kabul ediyoruz. Atiker Yazılım'ın geliştirdiği Clomosy, bu vizyonumuzun somut bir yansımasıdır. Clomosy, yalnızca bir mobil uygulama geliştirme platformu değil; bireylerin ve kurumların teknolojiyle olan bağlarını güçlendiren, onları dijital dönüşüm sürecine hazırlayan bir ekosistemdir.

Bu platform, teknolojiye erişimi kolaylaştırırken, aynı zamanda bireylerin kodlama gibi kritik bir beceriyi öğrenmelerine imkân tanır. Clomosy, yerli ve milli yapısıyla, kullanıcıların projelerini hızlı ve etkili bir şekilde geliştirmelerini sağlarken, aynı zamanda ülkemizin dijital bağımsızlık hedeflerine de katkı sunmaktadır. Yazılım geliştirmenin karmaşıklığını ortadan kaldırarak, herkesin fikirlerini hayata geçirebileceği bir zemin hazırlıyor.
Gelecek, teknolojiyi yalnızca kullanan değil, onu dönüştüren bireylerin ellerinde şekillenecektir. Clomosy, işte tam da bu noktada, eğitimden iş dünyasına kadar geniş bir yelpazede bir rehber görevi üstleniyor. Yapay zekâ destekli araçları, kişiselleştirilmiş öğrenme özellikleri ve platform bağımsız yapısıyla, her yaştan ve her seviyeden bireye eşsiz bir deneyim sunuyor. Bu deneyim, hem bireysel gelişimi hızlandırmakta hem de toplumsal kalkınmanın temel taşlarını oluşturmaktadır.

Bu kitap, teknolojiyle donanmış bireyler yetiştirmek ve onların gelecekte önemli roller üstlenmesine rehberlik etmek için hazırlandı. Clomosy ile kodlama öğrenmek, sorunlara çözüm üretmek ve dijital dünyanın sunduğu fırsatları en iyi şekilde değerlendirmek mümkün. Kitabın, teknolojiye dair perspektifinizi genişleterek sizlere ilham vereceğine inanıyorum.
Değişimden korkmadan, fırsatları değerlendirerek ve öğrenmeye açık bir şekilde ilerleyerek geleceği hep birlikte şekillendireceğiz.

OMA Holding
Yönetim Kurulu Başkanı
Ömer **ATİKER**

GİRİŞ

Dijital dönüşüm çağında teknoloji, bireylerin ve kurumların başarılarını yeniden tanımlıyor. Atiker Yazılım olarak, değişimin bu hızına ayak uydurabilmek ve daha güçlü bir gelecek inşa etmek için çalışıyor, yenilikçi projelerle teknolojiyle olan bağınızı güçlendirmeyi hedefliyoruz. Bu anlayışla geliştirdiğimiz **Clomosy**, bireylere ve kurumlara dijital dünyada güçlü bir yer edinmeleri için kapsamlı bir çözüm sunuyor.

Clomosy, herkesin kolayca kullanabileceği ve internet bağlantısıyla tüm işletim sistemlerinde erişim sağlayabileceği bir platform olarak tasarlandı. Kurulum ya da özel donanım gereksinimi olmadan, her ortamda projelerinizi geliştirmenize olanak tanıyor. Kullanıcı dostu yapısıyla, teknik deneyimi olmayanların dahi hızla projelerini hayata geçirebileceği bir sistem sunuyor.
Yazılım geliştirme süreçlerinde hızı ve esnekliği ön plana çıkaran Clomosy, yazılan kodların çıktısını tüm platformlarda anında kullanılabilir hale getiriyor. Ayrıca, TRObject dili ile sunulan bileşenler, iş süreçlerinde ve kişisel projelerde uygulama geliştirmeyi hızlı ve etkili bir hale getiriyor.

Yapay zekâ destekli asistanlar, kullanıcıların yazılım geliştirme sürecindeki hataları hızla tespit ederek yönlendirmelerde bulunuyor. Bunun yanı sıra, kişiselleştirilmiş öğrenme olanakları, her bireyin kendine özgü bir deneyim yaşamasını sağlıyor. Bu özellikler, yalnızca eğitim ve öğretim süreçlerini desteklemekle kalmıyor, aynı zamanda iş dünyasının dijital dönüşüm yolculuğunu da kolaylaştırıyor.

Clomosy, sürekli güncellenen altyapısı ve güçlü ekosistemi ile projelerinizin güvenle ilerlemesine olanak tanıyor. Teknolojiyi ve yazılım geliştirmeyi daha hızlı, erişilebilir ve etkili bir süreç haline getirerek, kullanıcıların fikirlerini uygulamaya dönüştürmesine imkan tanıyor.

Bu platform, bireysel becerilerinizi geliştirmenize yardımcı olurken, ülkemizin teknoloji alanında bağımsızlığını artırma hedeflerine de katkıda bulunuyor. Clomosy, yalnızca bir yazılım platformu olmanın ötesinde, kullanıcılarının dijital dünyada etkili bir rol oynamasına olanak tanıyan bir çözüm sunuyor.

Clomosy ile öğrenme ve gelişim yolculuğuna adım atarak, bugünün dünyasına değer katacak projeler geliştirme fırsatı bulacaksınız. Bu vizyonla yanınızda olmaktan gurur duyuyoruz.

Ayrıca, bu kitabın hazırlanmasına katkı sağlayan çalışma arkadaşlarıma içten teşekkür ederim. Onların özverili çalışmaları bu projeyi hayata geçirmemizi mümkün kıldı.

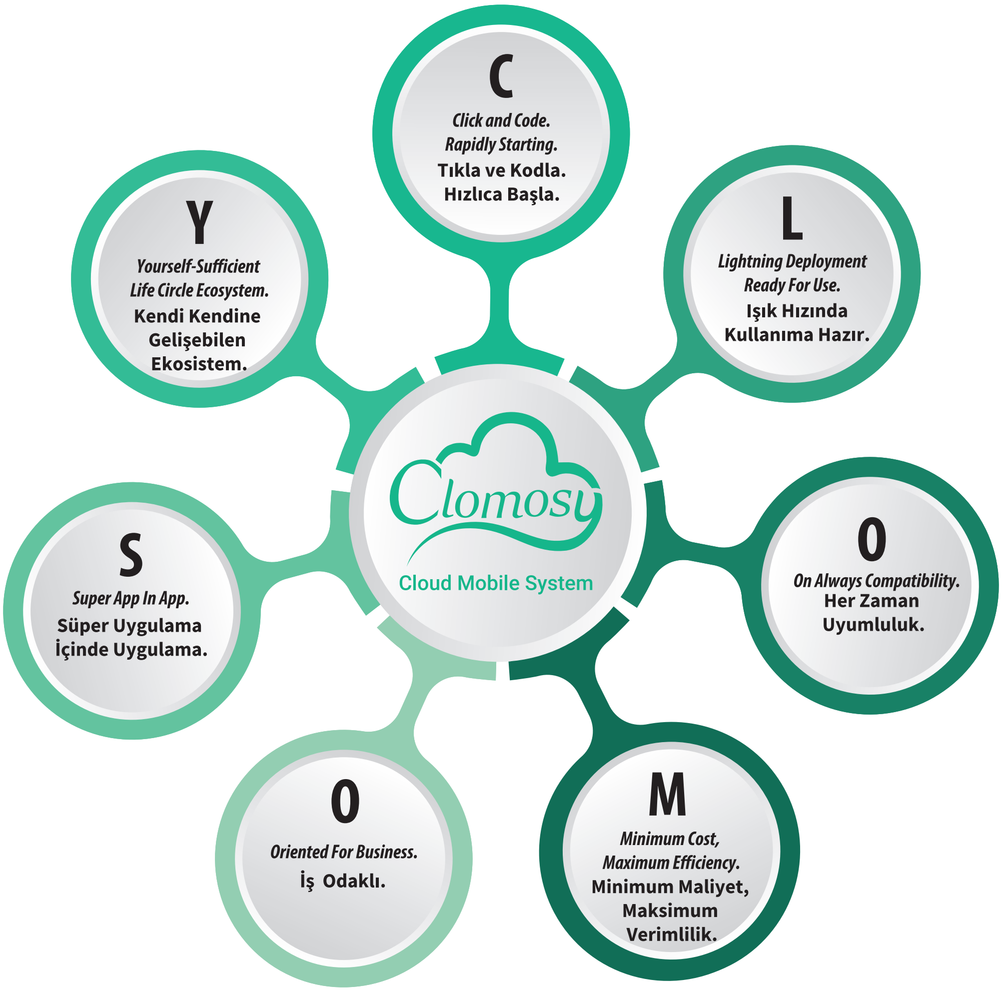
Atiker Yazılım Veri İşlem A.Ş.
Genel Müdür
Halil Özhan **SEZGİN**

BÖLÜM 1 : CLOMOSY GİRİŞ

CLOMOSY NEDİR?

Clomosy, bulut tabanlı bir mobil uygulama geliştirme platformudur. Geliştiricilerin web ortamında, çapraz platform (Android, Windows, iOS, macOS ve Linux) uygulama geliştirmelerine olanak sağlar. Clomosy ile geliştirilen uygulamalar bulutta saklanır ve uygulamaların verileri yerel ya da bulut ortamında tutulur.

Clomosy'nin Anahtar Özellikleri;



Görsel 1: Clomosy'nin Anahtar Özellikleri

C Tıkla ve Kodla. Hızlıca Başla **L** Işık Hızında Kullanıma Hazır **O** Her Zaman Uyumluluk
M Minimum Maliyet, Maksimum Verimlilik **O** İş Odaklı **S** Süper Uygulama İçinde Uygulama
Y Kendi Kendine Gelişebilen Ekosistem

CLOMOSY İLE PROGRAMLAMA

Bilgisayar programlama, programlama dillerini kullanarak bilgisayar yazılımları oluşturma süreci olarak tanımlanır. Tıpkı İngilizce, İspanyolca veya Fransızca konuşup anlamak gibi bilgisayarlar da belirli dillerde yazılmış programları anlayabilir. Bu dillere programlama dilleri denir. Başlangıçta sadece birkaç programlama dili mevcuttu. Bunları öğrenmek ve anlamak oldukça kolaydı. Ancak bilgisayarlar ve yazılımlar giderek daha karmaşık hâle geldikçe programlama dilleri hızla gelişti ve yol boyunca daha karmaşık kavramlar eklendi. Bu durum, insanları bilgisayar programlamayı öğrenme ve denemekten vazgeçirmeye başladı.

Clomosy, programlamaya yeni başlayanlar için çok daha kolay, erişilebilir ve eğlenceli hâle getirilmiş bir platformdur. Platformun amacı; bilgiye erişim ve öğrenme sürecindeki engelleri ortadan kaldırarak bilgisayar programlamanın ilgi çekici dünyasına bir köprü olmaktır. Clomosy platformu ile mobil ve masaüstü uygulamalar geliştirilebilir.

Clomosy'ye hızlı bir girişle başlayalım. Clomosy platformunda programlama yapabilmek için geliştirici hesabı, geliştirilen projeleri görebilmek için ise kullanıcı hesabı açılmalıdır. Hesapların nasıl oluşturulacağı aşağıdaki bölümde açıklanmaktadır.

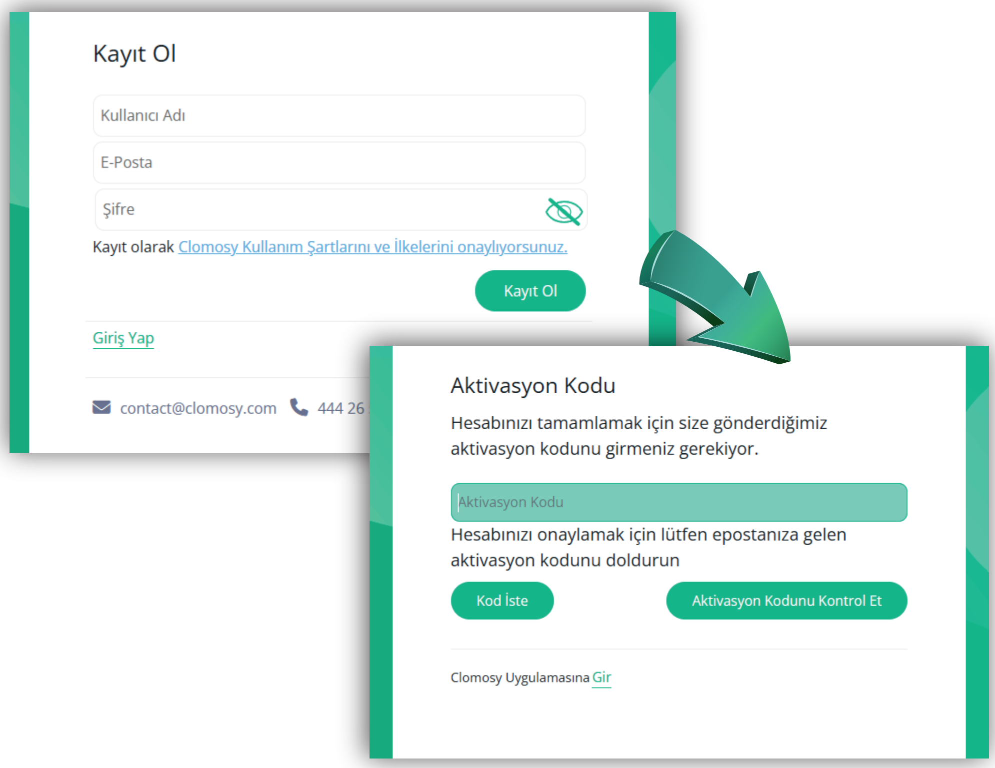
NOT:
Geliştirici (web hesabı) ve kullanıcı (mobil hesabı) olabilmek için 2 farklı e-posta hesabı ile kayıt işlemleri gerçekleştirilmelidir.

- Kullanılabilecek e-posta uzantıları; Gmail, iCloud, Yandex, Hotmail, Outlook veya '.tr' uzantılı adresler olabilir.

GELİŞTİRİCİ HESABI

Clomosy üzerinde programlama yapabilmek için oluşturulması gereken bir hesaptır. *https://clomosy.com.tr* adresinden "Kayıt Ol" butonuna tıklayarak aşağıdaki görselde yer alan sayfaya ulaşılır. Bu sayfada geliştirici için kullanıcı adı, e-posta adresi ve şifre girilmelidir.

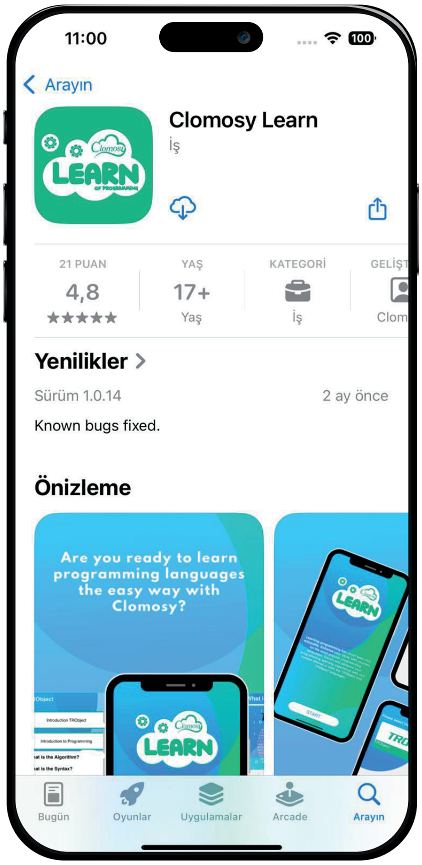
Veriler girilip "Kayıt Ol" butonuna tıklandıktan sonra e-posta adresine bir aktivasyon kodu gönderilecektir. Gelen aktivasyon kodu ilgili alana yazılıp onaylandığında giriş sayfasına yönlendirileceksiniz. Bu işlemden sonra geliştirici hesabınız oluşmaktadır.



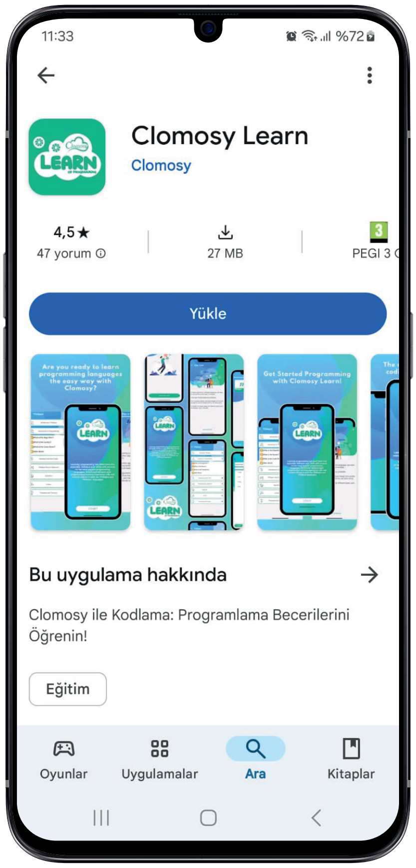
Görsel 2: Kayıt Ol ve Aktivasyon Kodu Ekranı

KULLANICI HESABI

Geliştirilen projeleri platform üzerinde görebilmek için kullanıcı hesabı açılmalıdır. Kullanıcı hesabı oluşturmak için App Store *(bkz. Görsel 3)* veya Play Store *(bkz. Görsel 4)* üzerinden Clomosy Learn uygulaması indirilmelidir.



Görsel 3: App Store



Görsel 4: Play Store

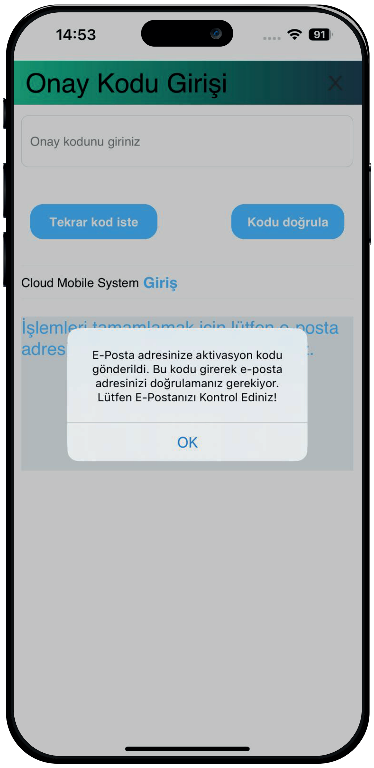
Uygulama ekranı açıldığında giriş ekranı gelecektir. Bu ekran üzerindeki “Kayıt Ol” butonuna tıklayarak “Hesap Oluşturma” sayfasına girilir. Bu sayfada kullanıcı için kullanıcı adı, e-posta adresi ve şifre girilir *(bkz. Görsel 5).*

Bilgiler girildikten sonra "Kayıt Ol" butonuna tıklanır ve e-posta adresine bir onay kodu gönderilir *(bkz. Görsel 6)*. Gelen onay kodunu girip onayladıktan sonra giriş sayfasına yönlendirilir. Bu işlemin ardından kullanıcı hesabı oluşturulmuş olur.

BİLGİLENDİRME:
Windows üzerinde Clomosy Learn uygulamasını kullanabilmek için https://clomosy.com/win/ClomosyLearn.zip adresinden Clomosy Learn uygulamasını indirmeniz gerekmektedir.



Görsel 5: Kullanıcı Hesabı Oluşturma



Görsel 6: Kullanıcı Hesabı Onay Kodu Giriş 1

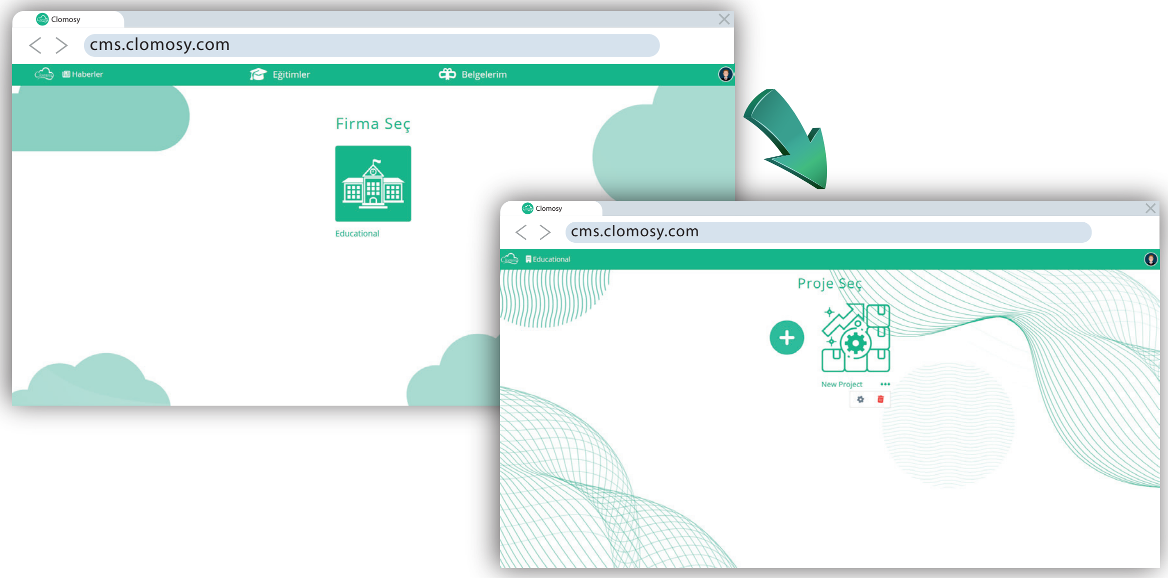


Görsel 7: Kullanıcı Hesabı Onay Kodu Giriş 2

PROJE OLUŞTURMA

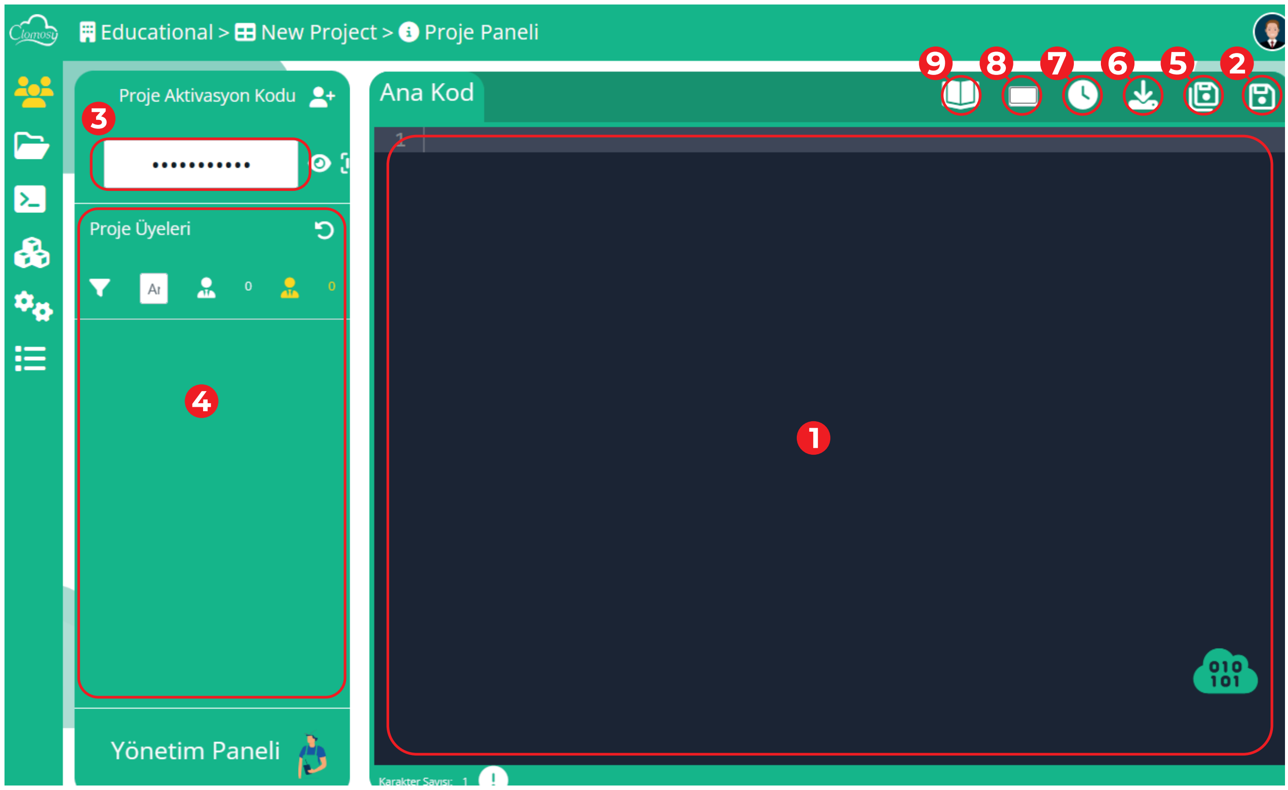
Oluşturulan geliştirici hesabına giriş yapıldığında aşağıdaki görselde bulunan “Firma Seç” sayfası ekrana gelir. Proje sayfasına geçiş yapabilmek için “Educational” firması seçilmelidir.

Proje oluşturmak için ekranın ortasında bulunan artı (+) butonuna tıklanmalıdır. Bir firma içerisinde birden fazla proje olabilir; proje adı girildikten sonra kaydedilebilir. Buna ek olarak daha sonra geliştirici dilediğinde proje ismini güncelleyebilir veya projesini silebilir. Bu işlemler için ise projenin adının yanında bulunan üç noktaya tıklamalıdır *(bkz. Görsel 8).*



Görsel 6: Firma Ekranı ve Proje Oluşturma

Sonrasında projeye tıklayarak geliştirici ortamına geçiş yapılabilir *(bkz. Görsel 9).*



Görsel 9: Geliştirici Ortamı

Bu ekran kodların yazılacağı ve çalıştırılacağı geliştirme ortamıdır. Bu ortamda birden fazla ayar yapılabilir. Kullanılması ve bilinmesi gereken bazı önemli noktalar vardır:

1. Kod geliştirme ortamıdır. Yazılan kodlar daha sonra değiştirilebilir veya kaydedip kullanılabilir.

2. Geliştirici ortamında yazılan kodların kaydedilmesi için kullanılan butondur.

3. Projeye üye (kullanıcı) eklemek için kullanılacak aktivasyon kodunu ifade eder. Bu kod sayesinde projeye birden fazla kullanıcı dâhil edilebilir.

4. Projedeki üyelerin listesi yer alır.

5. Proje içerisinde bulunan bütün birimlerin aynı anda kaydedilmesini sağlar.

6. “Ana Kod” sayfasını “.tro” dosyası olarak indirir.

7. Projede yazılan kodların otomatik olarak belirlenen süre aralığında kaydedilmesini sağlar. (Süre geliştirici tarafından istenilen zaman aralığında belirlenebilir.)

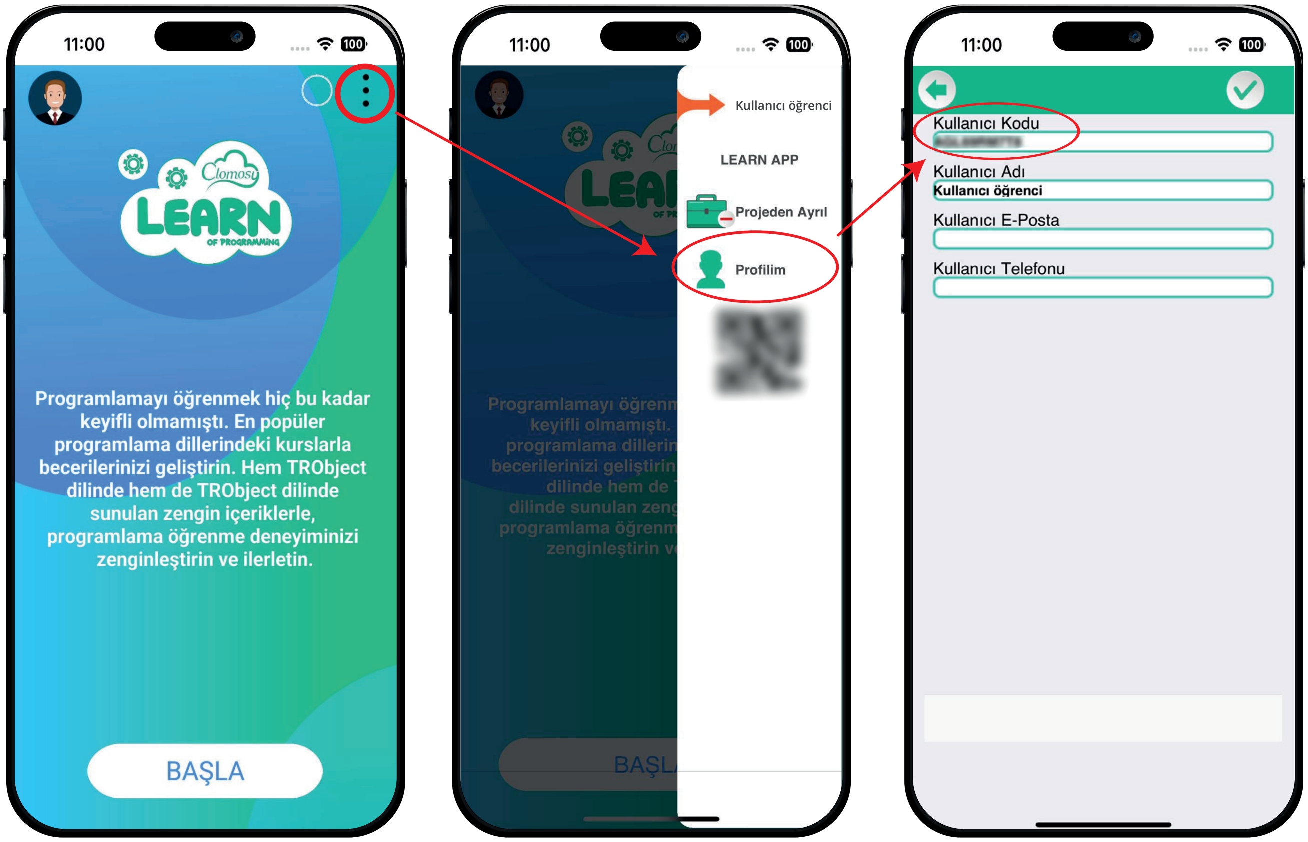
8. Renk paleti kullanılarak ekrana tıklandığında bir noktanın renk kodu elde edilir veya renk paletinden seçilen bir rengin kodu Hexadecimal, HSL veya RGB formatında alınır.

9. Geliştiriciyi dokümantasyon sitesine yönlendirir.
(https://www.docs.clomosy.com)

PROJEYE DÂHİL OLMAK

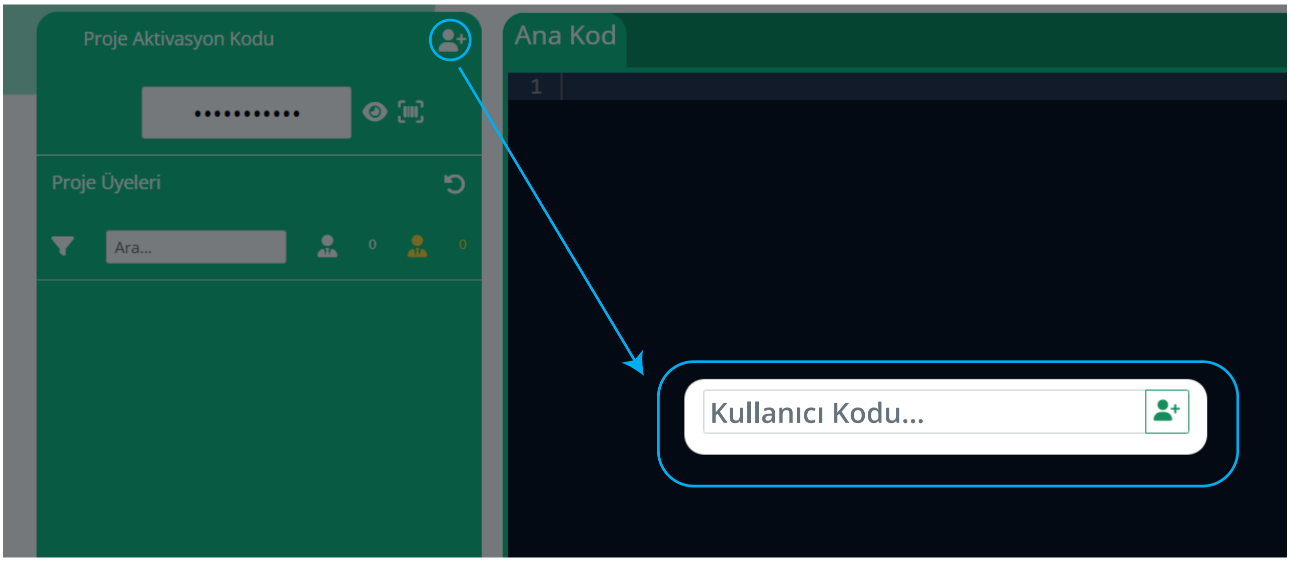
Clomosy'de geliştirilen projenin çıktısını görebilmek için kullanıcı, projeye üye olarak dâhil olmalıdır. Bunun için kullanıcı, geliştiriciye kullanıcı kodu bilgisini vermelidir.

Kullanıcı, Clomosy Learn uygulamasında hesap açıp giriş yaptığında, ekranın sağ üst köşesinde bulunan 3 noktaya tıklayarak 'Profilim' seçeneğinden 'Kullanıcı Kodu' bilgisine ulaşabilir *(bkz. Görsel 10)*.



Görsel 10: Kullanıcı Hesabı Kullanıcı Kod Bilgisi

Geliştirici tarafında, *Görsel 11*'de yer alan kutucuğa 'Kullanıcı Kodu' bilgi girişi yapılmalıdır. Bu şekilde kullanıcı projeye dâhil edilir.



Görsel 11: Geliştirici Ortamında Kullanıcı Member GUID Ekleme

BİLGİLENDİRME:

Yeni bir kullanıcı hesabı, geliştirici (web) ortamında 'Kullanıcı Kodu' ile projeye dâhil edildiğinde, mobil uygulamada projenin görülebilmesi için uygulamadan çıkıp tekrar girilmelidir. Bu şekilde proje listesi ekrana gelecektir.

Bu işleme, aynı kullanıcı yeni projeye dâhil olduğunda gerek duyulmaz. Mobil uygulamada proje listesinin yenilenmesi yeterlidir.

Alıştırmalar

1. Clomosy geliştirici hesabı oluşturma adımlarını sıralayınız.

2. Bir projeye üye eklemek için hangi bilgilerin gerekli olduğunu açıklayınız.

3. Clomosy'de projede yazılan kodların otomatik olarak kaydedilmesi için hangi buton kullanılır?

4. Clomosy'de bir projedeki tüm birimlerin aynı anda kaydedilmesi için hangi buton kullanılır?

5. Clomosy'nin 'Ana Kod' sayfasını bir dosya olarak indirilirken hangi dosya formatı kullanılır ve hangi butona tıklanmalıdır?

6. Proje ortamında bir kullanıcıyı projeye dâhil etmek için hangi bilgi gereklidir ve bu bilgiye nasıl ulaşılır?

7. Clomosy Learn uygulamasında yeni bir projeye dâhil olduktan sonra proje listesini görebilmek için ne yapılmalıdır?

8. Clomosy'de proje ismini değiştirmek veya projeyi silmek için hangi adımlar izlenmelidir?

https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum1

Ek Kaynaklar

- Clomosy Nedir?
  https://www.docs.clomosy.com/Clomosy

- Clomosy Mobil Uygulama Kurulumu
  https://www.docs.clomosy.com/Users

- Proje Oluşturma
  https://www.docs.clomosy.com/Project_Creation

- Clomosy Kullanımı Anlatımı
  https://www.youtube.com/watch?v=qwC8jDZaHRA

BÖLÜM 2 : İLK PROGRAMLAMA

İLK PROGRAMLAMA

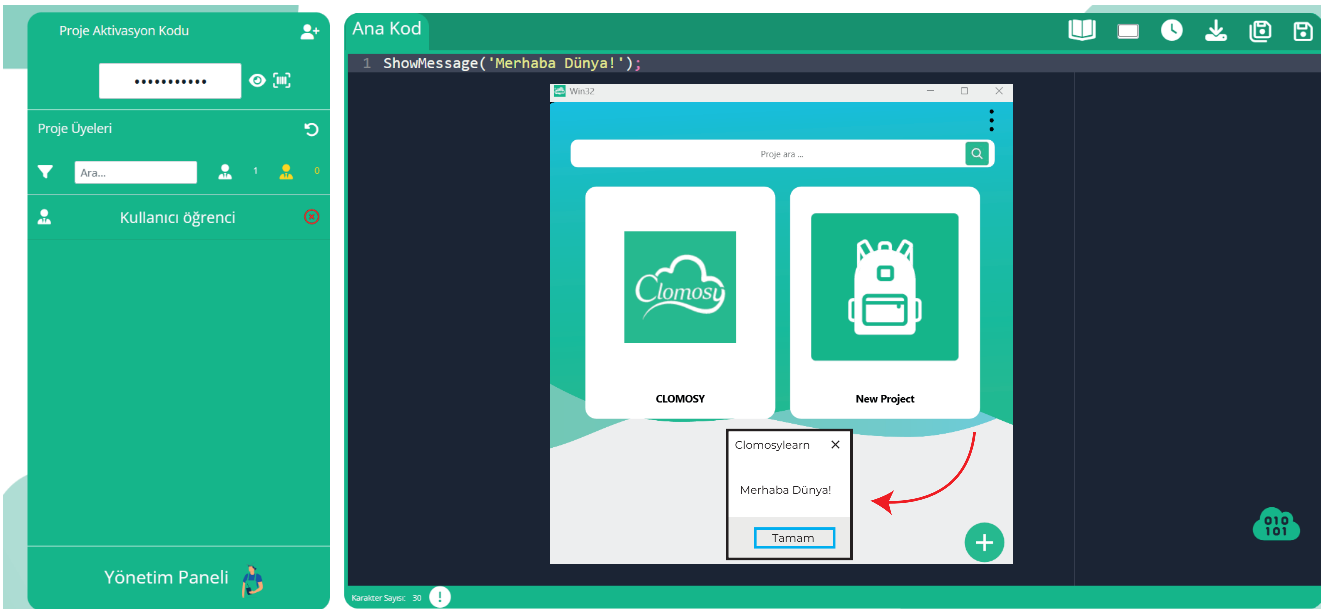
Bu aşamaya kadar Clomosy platformu tanıtıldı. Artık programlama sürecine geçilebilir. Daha önce belirtildiği gibi geliştirme ortamı kodların yazıldığı, projelerin geliştirildiği alandır. Şimdi geliştirme ortamında kod yazma adımlarına geçilecek ve *Görsel 12'*deki örnek kod satırı uygulanacaktır.



Görsel 12: Geliştirici Ortamı Kod Ekranı

Bu, Clomosy'de ilk uygulama örneğidir. Geliştirici ortamında (web) örneği çalıştırmak için sağ üst köşedeki 'Kaydet' butonuna tıklanmalıdır *(bkz. Görsel 12).*

Projeyi görüntülemek için mobil uygulama üzerinden proje adına tıklanmalıdır. Ve kod doğru yazıldıysa mesaj penceresi ekrana gelecektir *(bkz. Görsel 13).*



Görsel 13: Clomosy Learn Uygulama Sonucu

Konsol Nedir?

Konsol, kullanıcının komutlar girerek bilgisayarla etkileşimde bulunmasını sağlar ve bu komutların sonuçları yine metin olarak görüntülenir. Konsol hem sistem yönetimi hem de geliştirici araçları için yaygın olarak kullanılır.

'Merhaba Dünya!' yazısını konsol üzerinde görebilmek için kod, geliştirme ortamına yazılmalıdır *(bkz. Kod 1)*.

Ana Kod

```
Console.Text('Merhaba Dünya!');
```

Kod 1: İlk Uygulama Örneği

Clomosy Learn uygulaması Windows (.exe) üzerinde çalıştırılıp projeye tıklandığında konsol ekranına, metin gelecektir *(bkz. Görsel 14)*.



Görsel 14: Konsol Ekranı

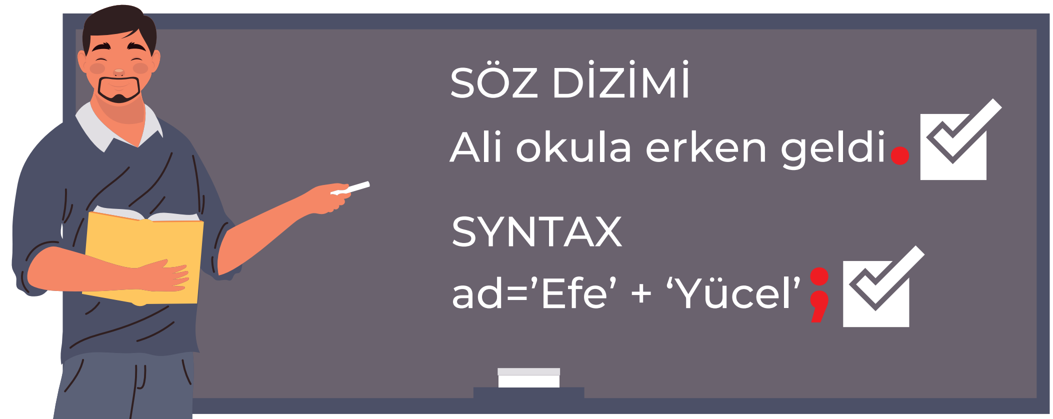
Tebrikler! Az önce Clomosy'de ilk program yazıldı ve çalıştırıldı. Çok küçük ve basit bir program ancak gerçek bir bilgisayar programcısı olma yolunda büyük bir adım! Şimdi, daha büyük programlar oluşturmak için devam edebiliriz.

CLOMOSY DİL YAPISI

Clomosy platformunda geliştirici ortamında kodlama yapabilmek için TRObject programlama dili kullanılır. Dil yapısına geçmeden önce söz dizimi ele alınmıştır.

Söz Dizimi (Syntax) Nedir?

Söz dizimi, programlama dillerinde komutların nasıl yazılması gerektiğini belirleyen kuralları ifade eder. Bu kurallar, bilgisayarın verilen talimatları doğru bir şekilde yorumlayıp çalıştırabilmesi için gereklidir. Tıpkı doğal dillerdeki dil bilgisi gibi programlama dillerinin de komutların doğru sıralanmasını ve yazımını düzenleyen bir yapısı vardır *(bkz. Görsel 15)*.



Görsel 15: Söz Dizimi Örneği

Günlük yaşamda nasıl ki cümlelerin anlamlı olması için dil bilgisi kurallarına uymamız gerekiyorsa programlama dillerinde de komutların işlevsel olabilmesi için belirlenmiş söz dizimi kurallarına uyulması zorunludur. Örneğin; Türkçe'de doğru bir cümle kurabilmek için özne, yüklem ve nesnenin belirli bir sıraya göre yerleştirilmesi gerekir.

Doğru Cümle:

Ali okula gitti.

Özne Ali, nesne okul, yüklem ise gitti olarak belirlenir. Bu cümledeki söz dizimi kuralları doğrudur ve anlamlı bir cümle oluşturur.

Yanlış Cümle:

Gitti Ali okula.

Bu cümlede kelimeler doğru bir sırayla yerleştirilmemiştir ve bu yüzden anlamı bozulmuştur. Bu, söz dizimi hatalarına benzer. Programlama dilinde de doğru söz dizimi kullanılmadığında, kod çalışmayabilir veya istenilen sonucu vermeyebilir. Örneğin; değişken tanımlaması yaparken mutlaka başına değişken olduğunu belirten “var” anahtar kelimesi yazılmalıdır. Ardından istenilen değişken adı, iki nokta (:) ve türü yazılmalıdır.

Doğru Yazım:

```
Var
  Ad : String;
```

Yanlış Yazım:

```
Var
  String : Ad;
```

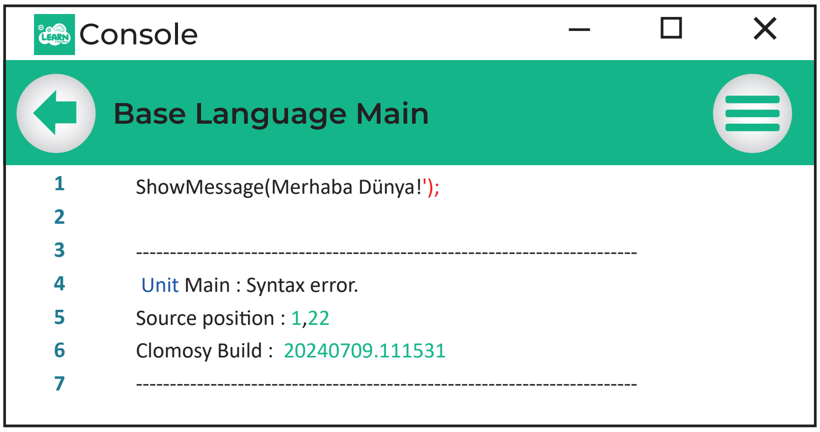
İlk program örneği üzerinde basit bir hata yapalım ve ekranda nasıl görünür buna bakalım. Örnekte, ShowMessage üzerinde yazılan metnin başındaki tırnak silinmeli ve proje tekrar çalıştırılmalıdır *(bkz. Kod 2)*.

Ana Kod

```
ShowMessage(Merhaba Dünya!');
```

Kod 2: İlk Uygulama - Hatalı Kod

Kod 2'de, söz diziminde bir hata yapıldı. Bu hata *Syntax error* olarak döner *(bkz. Görsel 16)*.



Görsel 16: Konsol Hata Ekranı

BİLGİLENDİRME:
Kod içerisinde hata yapıldığında *Görsel 16*'daki gibi konsol ile karşılaşılır. Gelen ekranda; öncelikle geliştirme ortamında yazılan kodlar ekrana gelir ve en alt kısımda hangi hata türünün alındığı ve bir altında ise kaynak kodda kaçıncı satırda hata olduğunu gösterir. Buna göre geliştirme ortamındaki hata satırını bulup istenilen hata düzeltilebilir.

Örnekte, 1.satırda *Syntax error* hatası oluştu. Bu hata tırnak (') açılmadığı için gerçekleşti. Hatanın giderilmesi için; metnin başına, eksik olan tırnak işareti eklenmeli, uygulama kaydedilip tekrar çalıştırılmalıdır.

Şimdi Clomosy'de dil yapısı nedir ona bakalım.

Clomosy platformu, TRObject dil yapısını kullanır. Bu yapı, farklı programlama paradigmalarına uygun tasarlanmış, geliştiricilere çeşitli olanaklar

TRObject Dil Yapısı:

- TRObject, C benzeri süslü parantezli bir dil yapısına sahiptir.
- Dosyaları ".tro" uzantısı ile tanımlanır.

Bu dil yapısı Clomosy platformunda esneklik ve özelleştirme sağlar. Yapının kullanımı; döngü, koşul, fonksiyon ve void tanımlamaları gibi konular içerisinde ele alınmıştır.

ALGORİTMA NEDİR?

Algoritma, belirli bir problemi çözmek veya belirli bir görevi yerine getirmek için izlenen adımlar dizisidir.

Örnek:
<u>Problem:</u> Boş bir kovayı su ile doldur ve çiçeği sula.

<u>Adımlar:</u>

1.Adım: Kovayı al.
2.Adım: Musluğu kontrol et, açık mı kapalı mı?
- Eğer musluk kapalıysa musluğu aç.

3.Adım: Musluğun altına kovayı yerleştir.
4.Adım: Su seviyesini kontrol et.
- Eğer su seviyesi istenilen düzeye ulaşmadıysa 5. adıma git.
- Eğer su seviyesi istenilen düzeye ulaştıysa 6. adıma git.

5.Adım: Kovayı suyla doldurmaya devam et sonra 4. adıma dön.
6.Adım: Musluğu kapat.
7.Adım: Kovayı musluğun altından al.
8.Adım: Çiçeğe git.
9.Adım: Çiçeği sulamaya başla.
10.Adım: Çiçeği sulamayı bitirdiğinde, kovayı yerine koy.

Bu günlük hayat algoritması, adım adım bir problemin nasıl çözüleceğini gösterir ve programlama dünyasında algoritmaların mantığını anlamak için güzel bir örnektir.

KOD BLOKLARI

Kod blokları, programlama dilinde belirli bir işlevi yerine getiren kod parçalarını gruplandırmak için kullanılır. Bu bloklar, genellikle belirli bir başlangıç, bitiş anahtar kelimesi veya sembolüyle çevrelenir. Bir arada yürütülmesi gereken komutları içerir.

CLOMOSY'DE KOD BLOKLARI

TRObject programlama dilinde kod blokları, TRObject söz dizimine göre süslü parantezler "{ ... }" arasında yer alır. Bu blok yapısı, belirli bir işlevi yerine getiren komutları gruplandırarak kodun düzenli ve okunabilir olmasını sağlar.

Kod blokları, ana kod bloğu dışında döngüler, koşullar, fonksiyon ve void gibi yapılarda da kullanılır. İlerleyen bölümlerde bu yapılar nasıl kullanılır gösterilecektir.

Bir kod bloğu, en basit haliyle şu şekilde yazılır *(bkz. Kod 3):*

Ana Kod

```
{
  // Bu alana kodlar yazılır (yorum satırı)
  ShowMessage('Merhaba Dünya!');
}
```

Kod 3: Kod Bloğu Kullanımı

YORUM SATIRLARI

Geliştirme ortamında *Kod 3*'te olduğu gibi mavi metinlerin görünmediğini fark ettiniz mi? Çünkü bu bir yorum satırıdır. Metnin önüne iki eğik çizgi (//) konularak yorum satırı açılır. Bilgisayar programı daha sonra bunu atlamayı ve bir sonraki kod satırına geçmeyi bilir.

Peki, yorum satırları ne için kullanılır? Profesyonel programcılar, kodlarını birbirlerine anlatmak için her zaman yorum satırlarını kullanır. Bu, ne yapıldığını açıklamanın bir yoludur. Ayrıca, bir süredir üzerinde çalışmayan bir koda geri dönüldüğünde de iyi bir hatırlatmadır.

Yazılan kodda bir çerçeve oluşturmak için de yorum satırları kullanılabilir. Bu, karmaşık bir görevin daha küçük görevlere bölünmesine ve planlanmasına yardımcı olur. Böylece büyük bir sorun yerine çok sayıda küçük sorun çözülebilir. " 'Merhaba Dünya' yazısının ardından 'Hoş geldiniz' ve 'Projenin sonuna geldik. Görüşmek üzere...' mesajı verilsin" demek kolay olabilir. *Kod 4*'te bilgisayardan istenen işlemin, açıklamalarla nasıl kullanılabileceğine dair bir örnek verilmiştir.

Ana Kod

```
{
  //İlk mesajımız Merhaba Dünya olsun.
  ShowMessage('Merhaba Dünya');
  //Ardından Hoş geldiniz yazısı ile iletişime geç
  ShowMessage('Hoş geldiniz.');
  //Projenin sonuna geldiğine dair mesaj ver
  ShowMessage('Projenin sonuna geldik. Görüşmek üzere...');
}
```

Kod 4: Yorum Satırı Kullanımı

Tekli yorum satırı açmanın yanı sıra çoklu yorum satırı da oluşturulabilir. Bu işlemi yapabilmek için yorum satırı başlangıcına eğik çizgi ve yıldız (/*), yorum satırının sonuna ise yıldız ve eğik çizgi (*/) konulmalıdır *(bkz. Kod 5)*.

Ana Kod

```
{
   /*
   1. İlk mesajımız Merhaba Dünya olsun.
   2. Ardından Hoş geldiniz yazısı ile iletişime geç
   3. Projenin sonuna geldiğine dair mesaj ver
   */
   ShowMessage('Merhaba Dünya');
   ShowMessage('Hoş geldiniz.');
   ShowMessage('Projemin sonuna geldik. Görüşmek üzere...');
}
```

Kod 5: Çoklu Yorum Satırı Kullanımı

Alıştırmalar

1. ShowMessage ve Console.Text komutları arasındaki farklar nelerdir?

2. Hangi durumlarda ShowMessage veya Console.Text komutları kullanılır?

3. Bir programlama dilinin söz dizimi hataları nasıl ortaya çıkar ve düzeltilir?

4. Aşağıdaki problemi adım adım çözmek için bir algoritma oluşturunuz?

    - Bir etkinlik düzenliyorsunuz ve bu etkinlik için belirli bir bütçeniz var. Etkinlikte yer alacak çeşitli aktiviteler var ve her bir aktivitenin bir maliyeti var. Amaç, bütçenizi aşmadan mümkün olduğunca fazla aktivite planlamaktır. (Bütçe oranını oluşturmuş olduğunuz algoritmaya göre belirleyin.)

https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum2

Ek Kaynaklar

- Kod Blokları Kullanımı
  https://www.docs.clomosy.com/Program_Structure

- Konsol Kullanımı
  https://www.docs.clomosy.com/Console_Applications

- Yorum Satırı Kullanımı
  https://www.docs.clomosy.com/Syntax

BÖLÜM 3 : DEĞİŞKENLER VE TÜRLER

DEĞİŞKEN NEDİR?

Değişken, bir değeri saklamak için kullanılan ve bu değere bir isim atanmış bellek alanıdır. Programların farklı parçaları arasında veri aktarımı yapmak, hesaplama sonuçlarını saklamak, kullanıcı girdilerini tutmak gibi çeşitli amaçlarla kullanılır. Değişkenler, programlama dillerinin temel yapı taşlarından biridir. Programların dinamikliğini, esnekliğini ve işlevselliğini sağlar.

Değişkenler kullanılmadan önce "var" anahtar kelimesi tanımlanmalıdır.

```
Var
        <degisken_adi> : <degisken türü>
```

BİLGİLENDİRME:

Değişken tanımlarken, türü açıkça belirtmek yaygın bir uygulamadır. Ancak, Clomosy'nin gelişmiş özelliklerinden biri olan **tür çıkarımı (type inference)** sayesinde, değişkenin türünü belirtmeden de tanımlama yapılabilir. Bu durumda, değişkenin türü atanan değere göre otomatik olarak belirlenir. Bu yöntemi kullanmak için 'var' anahtar kelimesinden sonra iki nokta üst üste (:) kullanmaya gerek yoktur. Örneğin:

```
var
  Sayi;
```

Bu tür tanımlama, kodun daha kısa ve okunabilir olmasını sağlar; ancak kod karmaşıklığı arttığında türleri açıkça belirtmek, kodun bakımı ve anlaşılabilirliği açısından faydalı olabilir.

DEĞİŞKEN ADLANDIRMA KURALLARI

Değişkenlerin, kendileriyle ilişkilendirilen adları vardır ve bu adlar belirli kurallara göre tanımlanmalıdır. Bu değişkenleri adlandırmak için bazı basit kurallar ve oldukça iyi yönergeler mevcuttur. Bunlar:

- İsim bir harfle başlamalı ve if, for, then, while, var, void, function vb. gibi anahtar kelimelerin hiçbiriyle aynı olmamalıdır.

- Bir ad, harf, rakam ve alt çizgilerin herhangi bir kombinasyonunu içerebilir.
- Değişken adlandırmasında birden fazla kelime kullanılacak ise arasında boşluk olmamalıdır.

Kitap Kodu ❌ **Kitap_Kodu** ✅

- Değişkenleri anlamlı bir şekilde adlandırmak önemlidir – amaçlarını açıklamak için uzun isimlendirilen değişken adları kullanılır.
- Değişken adlandırması büyük/küçük harfe duyarlı değildir. Bu sebepten "ad" isminde değişken ismi ile "AD" adında değişken ismi aynı şeyi ifade eder.
- Türkçe karakterler (ç, Ç, ğ, Ğ, İ, ö, Ö, ş, Ş, ü, Ü) kullanılmamalıdır. (Örn; ağaçİsmi yerine agac_ismi şeklinde olmalıdır.)
- Değişken tanımlamaları rakam ile başlamamalıdır.

Örneğin: Bir öğrencinin isminin tutulacağı değişkene ihtiyaç var ise değişkenin adı "OgrenciAdi" şeklinde olabilir.

```
Var
    OgrenciAdi
```

VERİ TÜRÜ NEDİR?

Veri türü, programlama dillerinde hangi türde veriyi (metin, sayı veya doğru-yanlış) saklayacağını belirleyen bir özelliktir. Farklı veri türleri, farklı bilgileri saklar ve bu bilgilerle yapılacak işlemleri etkiler.

Clomosy programlamada yaygın olarak kullanılan veri türleri bulunmaktadır *(bkz. Tablo 1)*.

Veri Türü	Açıklama	Kullanımı
String	Metin veya karakter dizilerini saklar.	Ad : String;
Integer	Tam sayıları saklar.	Kilo : Integer;
Boolean	Doğru (true) veya yanlış (false) değerlerini sağlar.	AktifMi : Boolean;
Float/Double	Ondalıklı sayıları saklar.	PuanOrtalamasi : Float;

Tablo 1: Değişken Türleri

Her değişken tipi, belirli bir türdeki veriyi saklamak için optimize edilmiştir. Bu sayede bellek yönetimi ve performans iyileştirilir. *Tablo 1*'deki veri türlerinden farklı olarak daha birçok özellik için kullanılan tür bulunmaktadır. Aşağıda, Clomosy'deki *String* ve *Integer* veri türleri detaylı incelenecektir.

METİNSEL VERİLERİ TUTMA

String, karakterlerden (harfler, sayılar, semboller vb.) oluşan bir veri türüdür. Metin veya karakter dizilerini değerini tutmak için kullanılır. Örneğin; bir kullanıcının adı, adresi veya iletişim bilgisi gibi verileri tutmak için *String* veri türü kullanılır.

String veri türünde tanımlanan bir değişken varsayalım. Bu değişkenin adını 'renkAdi' olarak adlandıralım ve veri türünü *String* olarak belirleyelim *(bkz. Görsel 17)*.

HATIRLATMA!
Değişken adı ve türünü belirlerken aralarında tanımlama için iki nokta (:) operatörü kullanılmalıdır.

```
Var
  renkAdi : String;
```



Görsel 17: Değişken Tanımlama

Tanımlanan değişkene değer atama işlemini yapabilmek için öncesinde belirtildiği gibi kod blokları açılmalı ve değer atama işlemi yapılmalıdır. Bir değişkene değer atamak için eşittir (=) işareti kullanılır.

```
renkAdi = 'Mor';
```

String türündeki bir değişkene değer atarken mutlaka tek tırnak ('…') sembolleri içerisinde yazılmalıdır *(bkz. Kod 6)*.

HATIRLATMA:
Değer, tek tırnak içerisine alınmaz ise string olmadığı anlamına gelir ve hata ile karşılaşılır.

Ana Kod

```
Var
  renkAdi : String;
{
  renkAdi = 'Mor';
}
```

Kod 6: Değişken Tanımlama ve Atama İşlemi



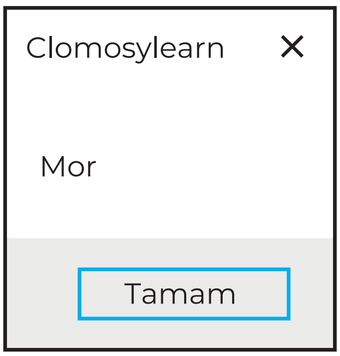
Görsel 18: Değişken Değer Ataması

Değişken oluşturuldu ve içerisine değer atandı. Şimdi bu değişken, kullanıcıya gösterilecektir. Bunu yapmak için ilk programda olduğu gibi *ShowMessage* fonksiyonu kullanılacaktır *(bkz. Kod 7).*

Ana Kod

```
Var
  renkAdi : String;
{
  renkAdi = 'Mor';
  ShowMessage(renkAdi);
}
```

Kod 7: Değişkenin ShowMessage ile Gösterimi 1



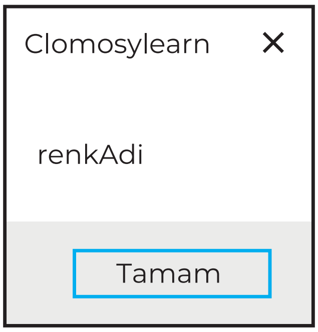
Görsel 19: Örnek Sonucu

Kod çalıştırıldığında mesaj penceresinde 'renkAdi' değişkeni içerisinde hangi değer bulunuyorsa o görüntülenecektir. *Görsel 19*' da olduğu gibi 'Mor' metnini verecektir. Eğer ki *ShowMessage* içerisinde değişken adı tırnak içerisinde yazıldıysa bilgisayar bunun bir metin olduğunu anlar ve değişken olarak algılamadan ekrana 'renkAdi' metnini yazdırır *(bkz. Görsel 20).* Bir de bu şekilde yazıldığında nasıl görüntülenir bakalım *(bkz. Kod 8).*

Ana Kod

```
Var
  renkAdi : String;
{
  renkAdi = 'Mor';
  ShowMessage('renkAdi');
}
```

Kod 8: Değişkenin ShowMessage ile Gösterimi 2



Görsel 20: Örnek Sonucu

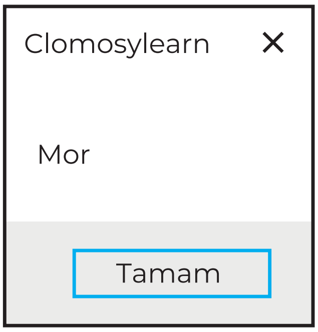
Daha önce *String* içerisine sayısal değerlerin metinsel olarak yazılabileceğinden bahsedilmişti. Şimdi *string* türünde sayısal bir renk kodu tutulacaktır. Tırnak içerisinde yazıldığı için bu değer string olarak kabul edilecek ve bilgisayar buna göre işlem yapacaktır *(bkz. Kod 9)*.

Ana Kod

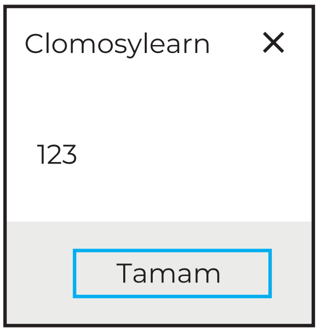
```
Var
  renkAdi, renkKodu : String;

{
  renkAdi = 'Mor';
  renkKodu = '123';
  ShowMessage(renkAdi);
  ShowMessage(renkKodu);
}
```

Kod 9: String Türünde Sayısal Metin Tanımlama



Görsel 21: String Türünde Sayısal Renk Adı



Görsel 22: String Türünde Sayısal Renk Kodu

Görüldüğü gibi tırnak içerisinde yazılı olan her değer, bilgisayarda bir string türü olduğu kabul edilir ve bu şekilde işleme alınır (*bkz. Görsel 21 ve Görsel 22).*

SAYISAL VERİLERİ TUTMA

Metinsel verileri saklamak için değişkenlerin nasıl kullanılabileceği ele alındı. Şimdi, değişkenlerdeki sayıların nasıl tutulacağı ve değiştirilebileceği incelenecektir.

Integer, tam sayıları (negatif, pozitif ve sıfır) saklayan bir veri türüdür. Matematiksel hesaplamalar ve sayısal verilerle çalışmak için kullanılır. Örneğin; bir sınıftaki öğrenci sayısı yada sepetteki elmanın ağırlık miktarı gibi bilgileri tutmak için *Integer* veri türü kullanılır.

Basit bir örnek ile programlamaya başlayalım:

Integer veri türünde tanımlanan bir değişken belirleyelim. Bir masanın boyunu tutabileceğimiz ‘Boy’ adlı değişkeni tanımlayalım *(bkz. Kod 10).*

Ana Kod

```
Var
  Boy : Integer;
```

Kod 10: Değişken Tanımlama

Tanımlanan değişkene değer atama işlemi yapabilmek için kod blokları açılmalı ve değer atama işlemi yapılmalıdır.

Ana Kod

```
Var
  Boy : Integer;
{
  Boy = 28;
}
```

Kod 11: Değişkene Değer Atama

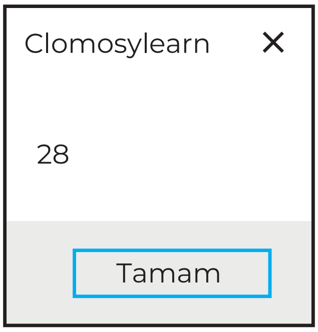
Değişken oluşturulup içerisine değer atandı. Şimdi bu değişken kullanıcıya gösterilmelidir *(bkz. Kod 12).*

Ana Kod

```
Var
   Boy : Integer;
{
   Boy = 28;
   ShowMessage(Boy);
}
```

Kod 12: Değişken Değerini Kullanıcıya Gösterme

Kullanıcı hesabında proje çalıştırıldığında çıktı *Görsel 23*'deki gibidir:



Görsel 23: Boy Adlı Değişken Sonucu

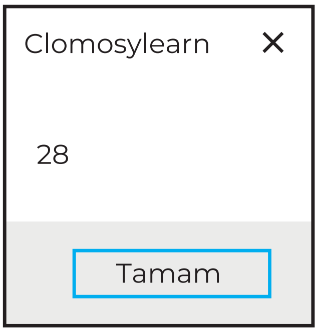
Programda sadece bir masanın boyu tutuldu ve ekranda gösterildi. Şimdi, örneğimizi değiştirelim ve 'Boy' değerine 3 ekleyelim. Ardından kodu tekrar çalıştıralım.

Ana Kod

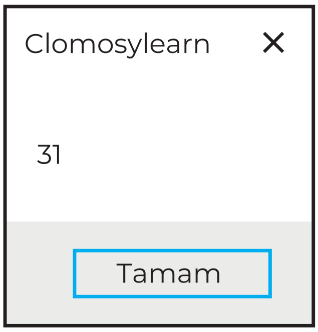
```
Var
   Boy : Integer;
{
   Boy = 28;
   ShowMessage(Boy);
   Boy = 28 + 3;
   ShowMessage(Boy);
}
```

Kod 13: Değişken Değerini Güncelleme

Örnekte masanın boyuna 3 eklenerek sonuç tekrar ekrana gösterildi *(bkz. Kod 13)*. 'Boy' değişkeninin değeri başlangıçta 28 iken, 3 eklendiğinde bu değer 31 oldu. *Görsel 24* ve *Görsel 25*'deki çıktı bu işlem sonucunda elde edildi.



Görsel 24: Örnek Sonucu 1



Görsel 25. Örnek Sonucu 2

Benzer şekilde sayılar çıkarılabilir, çarpılabilir veya bölünebilir.

Çıkarma işlemi:

```
Boy = Boy - 3;
```

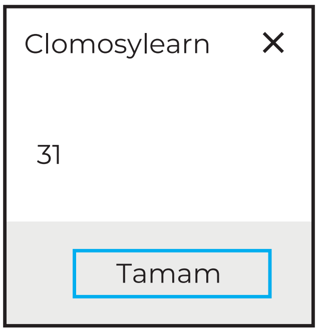
Şimdi bu örneği yeni oluşturulan bir değişkene atama işlemi ile yapalım. 'YeniBoy' adında bir değişken oluşturalım ve 'Boy' değişkeni içerisindeki değere 3 ekleyelim ve bu sonucu yeni değişkenimize aktaralım. Bu şekilde kullanıcıya gösterelim *(bkz. Kod 14)*.

Ana Kod

```
Var
   Boy, YeniBoy : Integer;
{
   Boy = 28;
   YeniBoy = Boy + 3;
   ShowMessage(YeniBoy);
}
```

Kod 14: Değişken Değerini Toplayarak Güncelleme

Görüldüğü gibi yine aynı sonuç ekrana geldi *(bkz. Görsel 26)*.



Görsel 26: YeniBoy Adlı Değişken Sonucu

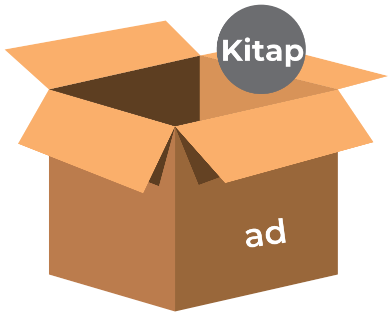
Uygulama

Şimdi bu öğrenilenler farklı bir örnek üzerinde incelenecektir.

Bir değişkenin adedi *(bkz. Görsel 27)* ve adı *(bkz. Görsel 28)* olduğunu varsayalım. Bunlar, içine değer konulan kutular olarak düşünülebilir.



Görsel 27: Uygulama Şekli 1



Görsel 28: Uygulama Şekli 2

Ana Kod

```
Var
  adet : Integer;
  ad : String;
{
  adet = 10;
  ad = 'Kitap';
}
```

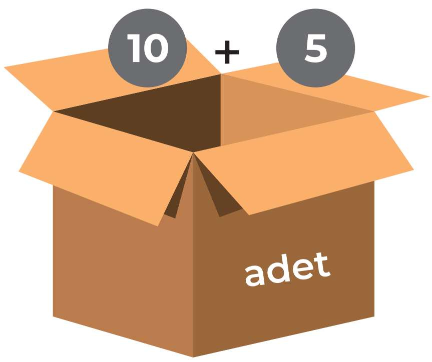
Kod 15: Uygulama Kodu

Bilgisayar önce ‘adet’ adlı değişkene 10 değerini atar, ardından ‘ad’ adlı değişkene ‘Kitap’ değerini atar *(bkz. Kod 15)*.

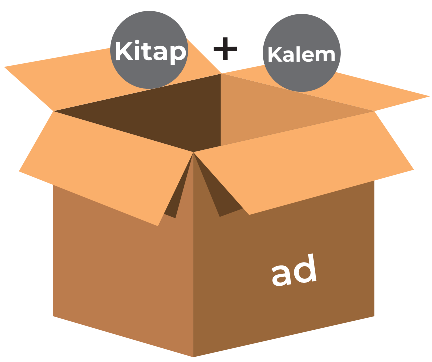
Farklı değişkenlerin, sayılar veya metinler gibi farklı veriye sahip olabileceği unutulmamalıdır.

Değişkenlere daha karmaşık değerler atanabilir.

```
adet = 10+5
ad = 'Kitap' + 'Kalem'
```



Görsel 29: adet Adlı Değişken



Görsel 30: ad Adlı Değişken

Soldaki değer, değişkene atanmadan önce değerlendirilir. Bu durumda, ‘adet’ değişkeni 15 (10+5) değerini alır, ‘ad’ değişkeni ise ‘KitapKalem’ değerini alır. “+” işareti, iki metin dizisini birleştirir. Bu işleme “dizi birleştirme” denir ve iki metni yan yana birleştirmek için kullanılır *(bkz. Kod 16).*

Ana Kod

```
Var
   adet : Integer;
   ad : String;
{
   adet = 10 + 5;
   ad = 'Kitap' + 'Kalem';
}
```

Kod 16: Uygulama Örneği Değer Değiştirme

TİP DÖNÜŞÜMLERİ



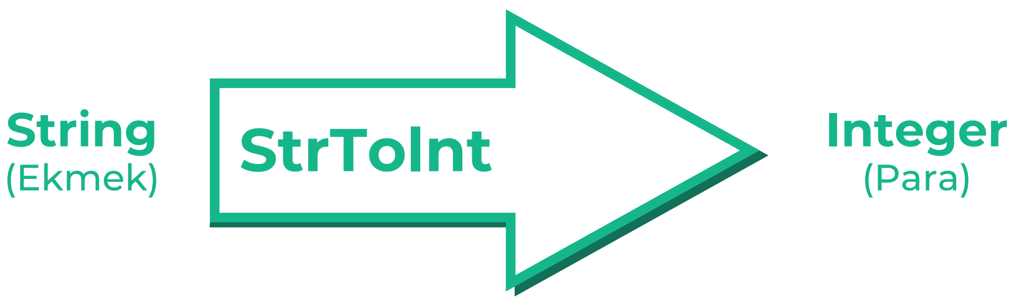
Görsel 31: Tip Dönüşümleri

Günlük hayatta fırıncı ekmeği paraya, müşteri ise parayı ekmeğe dönüştürür. Benzer şekilde, yazılımda da tip dönüşümleri vardır; bu bir veri türünün başka bir veri türüne dönüştürülmesi işlemidir. Programlama dillerinde bazı durumlarda veri türleri arasında dönüşüm yapmak gerekebilir.

Bir tam sayıyı ondalıklı bir sayıya dönüştürmek veya bir metni tam sayıya dönüştürmek gibi durumlar tip dönüşümlerine örnektir. Bu tür dönüşümler, programın gereksinimlerine ve mantığına bağlı olarak kullanılır.

StrToInt

StrToInt ile bir *string* veri türü, *integer* veri türüne dönüştürülür. Mesela fırıncı ekmeği paraya çevirmiştir *(bkz. Görsel 32).*



Görsel 32: StrToInt Dönüşümü

Ana Kod

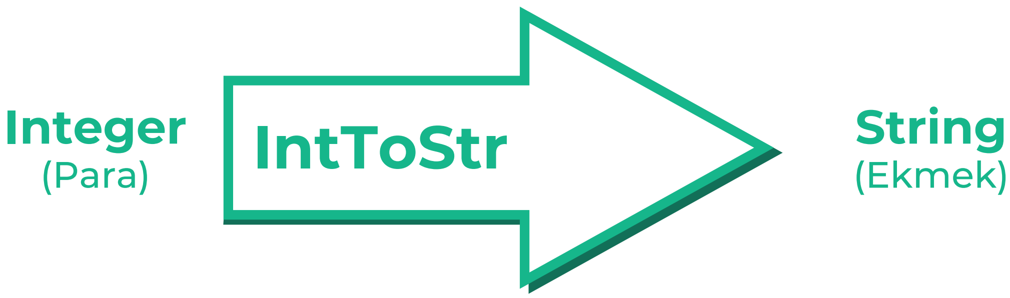
```
Var
   Param : String;
{
   Param = '70';
   ShowMessage(StrToInt(Param));
}
```

Clomosylearn
70
Tamam

Kod 17: StrToInt Kod Örneği

IntToStr

Integer veri türünü, *string* veri türüne dönüştürür *(bkz. Kod 18).*



Görsel 33: IntToStr Döüşümü

Ana Kod

```
Var
   Boy : Integer;
   Ad : String;
{
   Ad = 'Ali';
   Boy = 180;
   ShowMessage('Adım '+Ad+' Boyum '+IntToStr(Boy)+ ' cm');
}
```

Clomosylearn
Adım Ali Boyum 180 cm
Tamam

Kod 18: IntToStr Kod Örneği

BİLGİLENDİRME:
Metinsel verileri mesaj penceresinde *(ShowMessage)* gösterirken kelimeler arasında boşluk olması isteniyorsa kelimeler arasına '+' operatörü ile bir boşluk koyulabilir.

Alıştırmalar

QR OKUT

1. Yeni bir program yazın ve bir değişken oluşturunuz. Bu değişkenin adı 'sehirAdi' ve veri türü *String* olsun. 'sehirAdi' değişkenine bir değer atayınız ve bu değeri *ShowMessage* ile ekranda gösteriniz.

2. Değişkenin değerini değiştirmek için bir kod bloğu yazınız. Örneğin, 'sehirAdi' değişkenine ilk olarak Istanbul değerini atayınız ve daha sonra bu değeri Ankara olarak güncelleyiniz. Güncellenen değeri ekranda gösteriniz.

3. **Aşağıdaki değişkenleri tanımlayın ve uygun veri türlerini belirleyiniz:**
   - Bir öğrencinin yaşı (tam sayı), bir kitabın adı (metin)

   Her bir değişken için örnek değerler atayınız ve ShowMessage ile ekranda gösteriniz.

4. 'ilkSayi' ve 'ikinciSayi' adında iki tam sayı değişkeni oluşturunuz. Bu değişkenlere sırasıyla 10 ve 20 değerlerini atayınız. Bu iki sayının toplamını hesaplayın ve sonucu ShowMessage ile ekranda gösteriniz.

5. 'Ad' ve 'Soyad' adında iki metin değişkeni oluşturunuz. Bu değişkenlere isim ve soy isim değerlerini atayınız. Bu iki değişkenin değerlerini birleştirerek tam adı oluşturan 'AdSoyad' değişkenine atayınız ve ShowMessage ile ekranda 'Kullanıcı bilgisi = ad + soyad' olarak gösteriniz.

6. **Bir Sıcaklık Hesaplama Sorusu:**
   - **Soru:** Bir tarifte fırını 180°C'de ısıtmanız gerektiği belirtiliyor. Ancak fırınınız Fahrenheit cinsinden dereceleri gösteriyor. Fırınınızı hangi dereceye ayarlamalısınız?
   - **Açıklama:** Sıcaklık dönüşümünü uygulayarak Celsius'tan Fahrenheit'a geçiş yapılır. Formül: F = (C * 9/5) + 32.

7. **Bir Mesafe Hesaplama Sorusu:**
   - **Soru:** Bir koşu yarışmasında 5 kilometre koşmanız gerekiyor. Ancak antrenman yaptığınız parkur mil cinsinden ölçülmüş. Parkurunuzun kaç mil olduğunu bilmek istiyorsunuz. 5 kilometreyi mile çeviriniz.
   - **Açıklama:** Mesafe dönüşümünü uygulayarak kilometreden mile geçiş yapılır. 1 kilometre ≈ 0.621371 mil.

8. **Bir Hacim Hesaplama Sorusu:**
   - **Soru:** Bir içecek tarifi, 2 litre su eklemenizi istiyor. Ancak, ölçüm kabınız sadece mililitre cinsinden ölçümler gösteriyor. 2 litre su kaç mililitredir?
   - **Açıklama:** Hacim dönüşümünü uygulayarak litreyi mililitreye çeviriniz. 1 litre = 1000 mililitre.

9. **Bir Ağırlık Hesaplama Sorusu:**
   - **Soru:** Bir yemek tarifinde 500 gram un kullanmanız gerekiyor. Ancak, tartınız sadece ons cinsinden ölçüm yapabiliyor. 500 gram un kaç ons eder?
   - **Açıklama:** Ağırlık dönüşümünü uygulayarak gramı onsa çeviriniz. 1 gram ≈ 0.03527396 ons

10 **Bir Para Birimi Hesaplama Sorusu:**

- **Soru:** Yurt dışında tatildeyken 100 ABD Doları bozdurmanız gerekiyor. Ancak, bulunduğunuz ülkede para birimi Euro. Eğer 1 ABD Doları = 0.85 Euro ise, 100 ABD Doları kaç Euro eder?
- **Açıklama:** Para birimi dönüşümünü uygulayarak ABD Dolarını Euro'ya çeviriniz.

https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum3

Ek Kaynaklar

- Kodlama Hakkında
  https://www.docs.clomosy.com/TRObject_Language

- Tip Dönüşüm Kullanımı
  https://www.docs.clomosy.com/System_Library#Type_Conversion_Functions

BÖLÜM 4 : OPERATÖRLER



Operatörler, programlama dilinde çeşitli işlemleri gerçekleştirmek için kullanılan semboller ve işlevlerdir. Değişkenler, sabitler ve diğer ifadeler arasında çeşitli aritmetik, karşılaştırma, mantıksal ve atama işlemlerini gerçekleştirmek için kullanılır.

Operatör Türleri:

- Atama Operatörleri
- Aritmetik Operatörler
- İlişkisel Operatörler
- Mantıksal Operatörler

ATAMA OPERATÖRLERİ

Atama operatörü, programlama dilinde temel ve önemli bir talimattır. Atama sembolünün sağ tarafındaki hesaplanmış bir ifadenin değerini, atama sembolünün sol tarafındaki bir değişkene atar.

TRObject dil yapısında; eşittir (=) sembolü ile atama işlemi yapılır.

```
ad = 'Clomosy';
```

ARİTMETİK OPERATÖRLER

Aritmetik operatörler, matematiksel işlemleri gerçekleştirmek için kullanılan sembollerdir. Yazılımda bu operatörler, sayısal değerler üzerinde temel matematiksel işlemlerin yapılmasını sağlar.

Tablo 2'de aritmetik operatörler nedir ve ne amaçla kullanılır gösterilmektedir.

Operatör	Açıklama	Örnek
+	Toplama	x=6+7;
-	Çıkarma	x=5-3;
*	Çarpma	x=6*10;
/	Reel Sayı Bölümü	x=15/5;
Div	Tam Sayı Bölümü	x=10 div 4; // (x=2)
Mod	Mod Alma(Kalan)	x=28 mod 9; // (x=1)
^	Üs Alma	x=2^3; // (x=8)

Tablo 2: Aritmetik Operatörler

Kod 19'da iki değişken (a ve b) için *Tablo 2*'de yer alan operatör örnekleri verilmiştir.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum4/Ornek1)

Ana Kod

```
Var
   a, b, Toplama, Fark, Carpma, Bolme, Kalan, Bolum, UsAlma : Integer ;
{
   a = 10 ;
   b = 5 ;
   Toplama = a+b ;
   Fark = a-b ;
   Carpma = a*b ;
   Bolme = a/b ;
   Kalan = a mod b ;
   Bolum = a div b ;
   UsALma = a ^ b ;
   ShowMessage('Toplama sonucu : '+ IntToStr(Toplama)); //15
   ShowMessage('Çıkarma sonucu : '+ IntToStr(Fark)); // 5
   ShowMessage('Çarpma sonucu : '+ IntToStr(Carpma)); // 50
   ShowMessage('Bölme sonucu : '+ IntToStr(Bolme)); // 2
   ShowMessage('Bölme işleminde kalan sonucu : '+ IntToStr(Kalan)); //0
   ShowMessage('Bölme işleminde bölüm sonucu : '+ IntToStr(Bolum)); //2
   ShowMessage('Üs alma sonucu : '+ IntToStr(UsAlma)); //100000
}
```

Kod 19: Aritmetik Operatörler Kod Örneği

İLİŞKİSEL OPERATÖRLER

İlişkisel operatörler, iki değeri karşılaştırmak için kullanılır. Karşılaştırmanın sonucu doğru (TRUE) veya yanlış (FALSE) olmak üzere iki sonuçtan biri olarak elde edilir *(bkz. Tablo 3)*.

Operatör	Açıklama	Örnek
==	Eşittir	(21==51)//False
<>	Eşit Değil	(2<>3)//True
not	Değil İfadesi	not(21==10)//True
<	Küçük ise	(30<21)//False
>	Büyük ise	(3>4)//False
<=	Küçük veya Eşit ise	(1<=7)//True
>=	Büyük veya Eşit ise	(2>=8)//False

Tablo 3: İlişkisel Operatörler

Örnek

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum4/Ornek2)

Ana Kod

```
Var
  a, b : Integer ;
{
  a = 18 ;
  b = 7 ;
  ShowMessage (a == b); //False
  ShowMessage (a <> b); //True
  ShowMessage (not(a == b)); //True
  ShowMessage (a < b); //False
  ShowMessage (a > b); //True
  ShowMessage (a <= b); //False
  ShowMessage (a >= b); //True
}
```

Kod 20: İlişkisel Operatörler Kod Örneği

Elimizde iki kutu var ve bu kutuların boyutlarının eşit olup olmadığı kontrol edilmek isteniyor. Bu kutuların boyutlarını karşılaştırarak eşitlik kontrolü yapılabilir.

<u>Kutu Bilgileri:</u>

Kutu 1:
    Boyut: 30 cm
Kutu 2:
    Boyut: 30 cm

Bu senaryo bir kod örneği ile gösterilmek istendiğinde *(bkz. Kod 21);*

1. kutu için 'Kutu1Boyut' ve 2.kutu için 'Kutu2Boyut' değişkenleri *Integer* veri türünde oluşturulmalıdır. Değişkenler tanımlandıktan sonra değer atamaları yapılır. Eşit olduklarını göstermek için her iki değişkene de aynı sayılar yazılır.

İki kutunun boyutlarının eşit olup olmadığını belirlemek için *boolean* türünde 'BoyutlarEsit' adında bir değer tanımlanmıştır.

BOOLEAN DEĞERLER:
Boolean, mantıksal (lojik) ifadeleri temsil eden ve yalnızca iki olası değerden birine sahip olan bir veri türüdür. Bu iki değer genellikle **True** (doğru) ve **False** (yanlış) olarak adlandırılır. Boolean değerler, programlamada sıklıkla koşullu ifadeler ve kontrol yapılarında kullanılır.

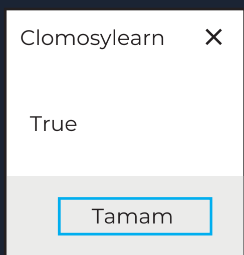
Son olarak eşitlik değerinden dönen mantıksal değer, *ShowMessage* ile gösterildiğinde her iki değer birbirine eşit olduğu için *True* değeri döndürecektir.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum4/Ornek3)

Ana Kod

```
Var
  Kutu1Boyut, Kutu2Boyut : Integer;
  BoyutlarEsit : Boolean;
{
  Kutu1Boyut = 30;
  Kutu2Boyut = 30;
  BoyutlarEsit = (Kutu1Boyut == Kutu2Boyut);
  ShowMessage(BoyutlarEsit);
}
```



Kod 21: Kutu Örneği

MANTIKSAL OPERATÖRLER

Mantıksal operatörler, koşul ifadelerinde kullanılır ve sonuç olarak True (doğru) veya False (yanlış) değeri döndürür. *Tablo 4*'te operatör çeşitleri incelenecektir:

Operatör	Açıklama	Örnek
&&	Mantıksal ve	(x=15)&&(y=7)
||	Mantıksal veya	(X=5) || (y=12)

Tablo 4: Mantıksal Operatörler

Mantıksal operatörler, günlük hayatta çeşitli durumları kontrol etmek ve kararlar almak için kullanılır.

&& (ve):

- İki veya daha fazla koşulun hepsi doğru olduğunda, sonuç doğrudur.
- Örneğin: Hem yağmur yağıyorsa hem de şemsiyeniz varsa dışarı çıkılabilir.

|| (veya):

- İki veya daha fazla koşuldan en az biri doğru olduğunda, sonuç doğrudur.
- Örneğin: Kahve içmek isteniyorsa ya da çay içmek isteniyorsa mutfağa gidilmelidir.

Günlük hayattan örneklerle açıklayalım:

&& (ve) Operatörü

Senaryo:
Bir arkadaşınızla dışarıda buluşmak istiyorsunuz. Dışarı çıkmak için hem hava güzel olmalı hem de işlerinizi bitirmiş olmalısınız.

Koşullar:

- Hava güzel mi? (HavaGuzel)
- İşler bitti mi? (islerBitti)

Mantıksal İfade:

QR Okut

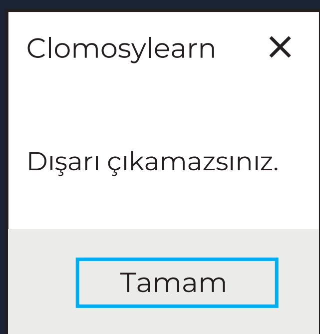
(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum4/Ornek4)

Ana Kod

```
var
  HavaGuzel, islerBitti, disariCikabilir: Boolean;
{
  HavaGuzel = True;  // Hava güzel
  islerBitti = False; // İşler bitmemiş

  disariCikabilir = (HavaGuzel && islerBitti);

 if disariCikabilir
    ShowMessage('Dışarı çıkabilirsiniz.');
 else
    ShowMessage('Dışarı çıkamazsınız.');
}
```



Kod 22: Ve (&&) Operatörü Örneği

Sonuç:
Hava güzel olsa bile işleriniz bitmediği için dışarı çıkamazsınız *(bkz. Kod 22)*.
Soru:
Kod 22'de "Dışarı çıkabilirsiniz." sonucunu gösteren algoritmayı tasarlayın.

|| (veya) Operatörü

Senaryo:
Bir film izlemek istiyorsunuz. Ya bir arkadaşınız size katılmalı ya da içecek olmalı.
Koşullar:
- Arkadaş katılıyor mu? (arkadasGeliyor)
- İçecek var mı? (icecekVar)

Mantıksal İfade:

QR Okut

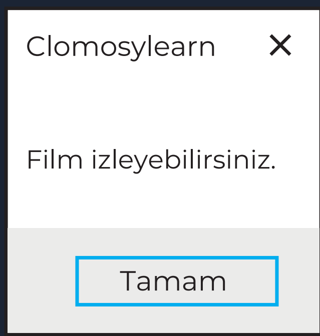
(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum4/Ornek5)

Ana Kod

```
var
 arkadasGeliyor, icecekVar, filmIzleyebilir: Boolean;
{
 arkadasGeliyor = False; // Arkadaş gelmiyor
 icecekVar = True;  // içecek var

 filmIzleyebilir = (arkadasGeliyor || icecekVar);

 if filmIzleyebilir
   ShowMessage('Film izleyebilirsiniz.');
 else
   ShowMessage('Film izleyemezsiniz.');
}
```



Kod 23: Veya (||) Operatör Örneği

Sonuç:
Arkadaşınız gelmese bile, içecek olduğu için film izleyebilirsiniz.

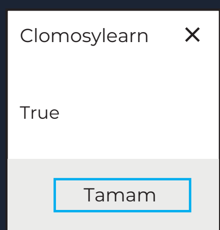
Bu örnekler, mantıksal operatörlerin günlük hayatta nasıl kullanılabileceğini ve karar alma süreçlerinde nasıl yardımcı olabileceğini göstermektedir.

Örnek

Ana Kod

```
Var
   x: Integer;

{
   x=18;
   ShowMessage(((x < 20)||(x > 10)));
}
```



Kod 24: Mantıksal Operatör Örneği

Kod 24'te, 'x' değişkeni tanımlanır ve değeri 18 olarak atanır. *ShowMessage* fonksiyonu, 'x' değişkeninin 20'den küçük veya 10'dan büyük olup olmadığını kontrol eder ve bu koşulda doğru olduğu için ekranda 'True' sonucu gösterilir.

Alıştırmalar

1. Matematik dersinizde kullandığınız ve bizim listelemediğimiz herhangi bir matematiksel operatör sizce var mıdır?
2. İki tane sayı yazıp bu sayıların toplamını, farkını, çarpımını ve bölümünü matematiksel operatörler yardımıyla sırasıyla ekranda gösteriniz.
3. İki mantıksal değer (True veya False) girilmesi istenen ve her iki değerin de doğru (True) olup olmadığını kontrol eden bir algoritma tasarlayınız.
4. İki sayıyı karşılaştırarak hangisinin büyük olduğunu gösteren bir kod yazınız.
5. Bir sayının çift mi tek mi olduğunu kontrol eden bir kod yazınız.
6. İki sayı belirtip sayılardan herhangi biri pozitifse mesaj veren bir kod yazınız.
7. Bir okulda kütüphaneye giriş yapabilmek için öğrencilerin kimlik kartlarını göstermesi veya öğretmenlerden izin alması gerekiyor. Aşağıda verilen duruma göre, kütüphaneye girip giremeyeceğinizi kontrol eden bir program yazınız. Program, öğrencinin kimlik kartı olup olmadığı ve öğretmen izni durumuna göre ekrana uygun mesajı yazdıracaktır.

Durum:

- Kimlik kartı: Hayır
- Öğretmen izni: Evet

Beklenen Sonuç:
Program, bu duruma göre "Kütüphaneye girebilirsiniz." veya "Kütüphaneye giremezsiniz." mesajlarından birini ekrana yazdırmalıdır.

https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum4

Ek Kaynaklar

- Operatör Kullanımı
  https://www.docs.clomosy.com/Operators

BÖLÜM 5 : KOŞULLAR

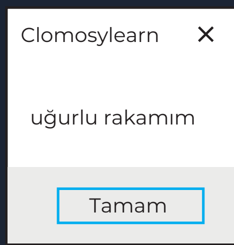
Günlük hayatta koşullara bağlı olarak farklı kararlar alınır. Örneğin, hava yağmurlu olduğunda dışarı çıkarken şemsiye alınır. Bilgisayarlar da benzer şekilde koşullara dayalı kararlar verebilir ve bu, programlamada belirli şartlar kullanarak yapılır.

Bilgisayarlara bir dizi talimat verebilir ve bu talimatları yerine getirmeleri sağlanabilir. Ancak, her program çalıştırıldığında aynı talimatların uygulanması istenmeyebilir. Örneğin; belirli bir sayı girildiğinde özel bir mesajın görüntülenmesi isteniyorsa bu durumda koşullu ifadeler kullanılarak belirli bir işlem gerçekleştirilir.

Eğer girilen sayı 1 ise program “uğurlu rakamım” mesajını görüntüler. Ancak, ‘Sayi’ değişkenine başka bir değer atanırsa bu mesaj görüntülenmez.

Ana Kod

```
Var
   Sayi : Integer;
{
  Sayi = 1;
  if Sayi == 1
     ShowMessage('uğurlu rakamım');
}
```



Kod 25: Koşul Örneği

Koşullar ve Karar Kontrol Yapıları

Koşullar, doğru (True) veya yanlış (False) olarak değerlendirilen ifadelerdir. Örneğin, "ikiz kız kardeşim var" ifadesi bir koşuldur. Eğer gerçekten bir ikiz kardeşiniz varsa bu koşul doğru (True), yoksa yanlış (False) olur.

Karar kontrol yapıları, yazılım programlarının akışını yönetmekte kritik bir rol oynar. İşte karar kontrol yapılarının yazılımda neden kritik olduğuna dair bazı temel nedenler:

- **Program akışını değiştirirler:** Koşullara göre programın farklı yollar izlemesini sağlarlar.
- **Dinamik yapı kazandırırlar:** Belirli koşulların doğru ya da yanlış olmasına göre farklı kod blokları çalıştırılır.
- **Esnek ve işlevsel yapı sağlarlar:** Programın daha dinamik ve kullanıcı girdilerine uygun şekilde çalışmasına imkân tanırlar.

Clomosy Üzerinde Kullanılan Karar Kontrol Yapıları

TRObject programlama dilinde kullanılan başlıca karar kontrol yapıları *Tablo 5*'te yer almaktadır.

Karar Kontrol Yapısı
If Deyimi
If – Else Deyimi
If – Else If Deyimi
Case Deyimi

Tablo 5: Karar Kontrol Yapısı

Bu karar kontrol yapıları, programların dinamik davranışlarını sağlar ve belirli koşullara göre farklı işlemleri gerçekleştirmelerine olanak tanır. Bu da programların daha esnek ve işlevsel olmalarını sağlar.

KOŞUL İFADELERİ

IF DEYİMİ

Programın belirli bir koşulu değerlendirmesini ve bu koşula bağlı olarak farklı kod bloklarını yürütmesini sağlar.

If deyimi, belirli bir koşulun doğru veya yanlış olup olmadığını kontrol eder ve doğru olduğunda belirli bir kod bloğunu çalıştırır.

Bir if deyimi genellikle şu şekilde yapılandırılır:

```
if  (koşul)
{
  Koşul doğruysa yapılacak işlemler
}
```

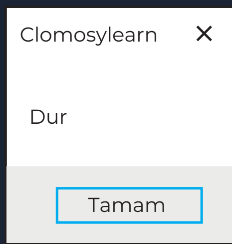
NOT:
if blokları için süslü parantez {...} kullanılır. Bir koşul bloğunda tek satır var ise süslü parantez {...} kullanılmak zorunlu değildir.



Örneğin: Yolda giden bir araba için koşul cümlesi şu şekilde yazılabilir: “Eğer kırmızı ışık yanarsa araba dursun.” *(bkz. Kod 26).*

Ana Kod

```
Var
  Lamba : String;
{
  Lamba = 'kırmızı';
  if (Lamba == 'kırmızı')
    ShowMessage('Dur');
}
```

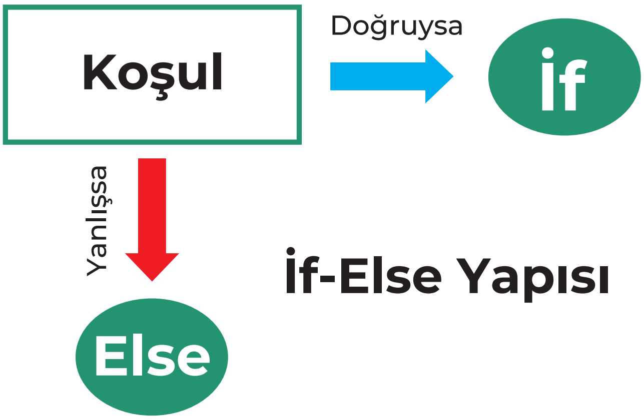


Kod 26: If Deyimi Örneği

IF – ELSE DEYİMİ

If – else yapısı *if* deyimi ile benzerdir ancak alternatif bir yol sunar. *If-else* deyimi şu şekilde yapılandırılır:

```
if (koşul)
{
  Koşul doğruysa yapılacak işlemler;
 }
else
{
   Koşul yanlışsa yapılacak işlemler;
}
```

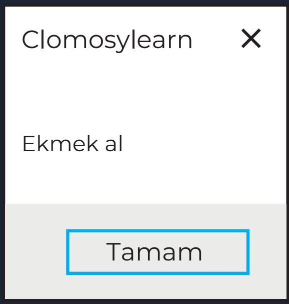


Görsel 34: If – Else Yapısı Diyagramı

Örnek 1

Ana Kod

```
Var
  Para : String;
{
  Para = 'param var';
  if (Para == 'param var')
  {
    ShowMessage('Ekmek al');
  }
  else
  {
    ShowMessage('Eve gel');
  }
}
```



Kod 27: If – Else Deyimi Örneği 1

Programın başlangıcında, 'Para' adında bir metin (String) değişkeni tanımlanır. Bu değişkene "param var" değeri atanır. Program, 'Para' değişkeninin değerinin "param var" olup olmadığını kontrol eder. Eğer 'Para' değişkeni "param var" değerine sahipse ekranda "Ekmek al" mesajı gösterilir. Eğer 'Para' değişkeni "param var" değerine sahip değilse ekranda "Eve gel" mesajı gösterilir *(bkz. Kod 27).*

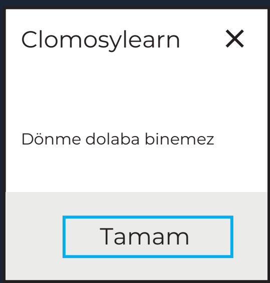
Örnek 2

Ana Kod

```
Var
  Boy : Integer;

{
  Boy = 110;

  if (Boy > 120)
    ShowMessage('Dönme dolaba binebilir');
  else
    ShowMessage('Dönme dolaba binemez');
}
```



Kod 28: If – Else Deyimi Örneği 2

Örnekteki kod, bir kişinin boyunu kontrol ederek dönme dolaba binip binemeyeceğine karar verir. ‘Boy’ adında bir tam sayı (Integer) değişkeni tanımlanır ve bu değişkene 110 değeri atanır. If deyimi kullanılarak ‘Boy’ değişkeninin 120’den büyük olup olmadığı kontrol edilir. ‘Boy’ değişkeninin değeri 120’den büyükse ekranda “Dönme dolaba binebilir” mesajı gösterilir. ‘Boy’ değeri 120’den küçük veya 120’ye eşitse else bloğu çalıştırılır ve “Dönme dolaba binemez” mesajı gösterilir. Bu adımlar kişinin boyuna göre dönme dolaba binip binemeyeceğini belirlemek amacıyla uygulanır (*bkz. Kod 28).*

IF – ELSE IF DEYİMİ

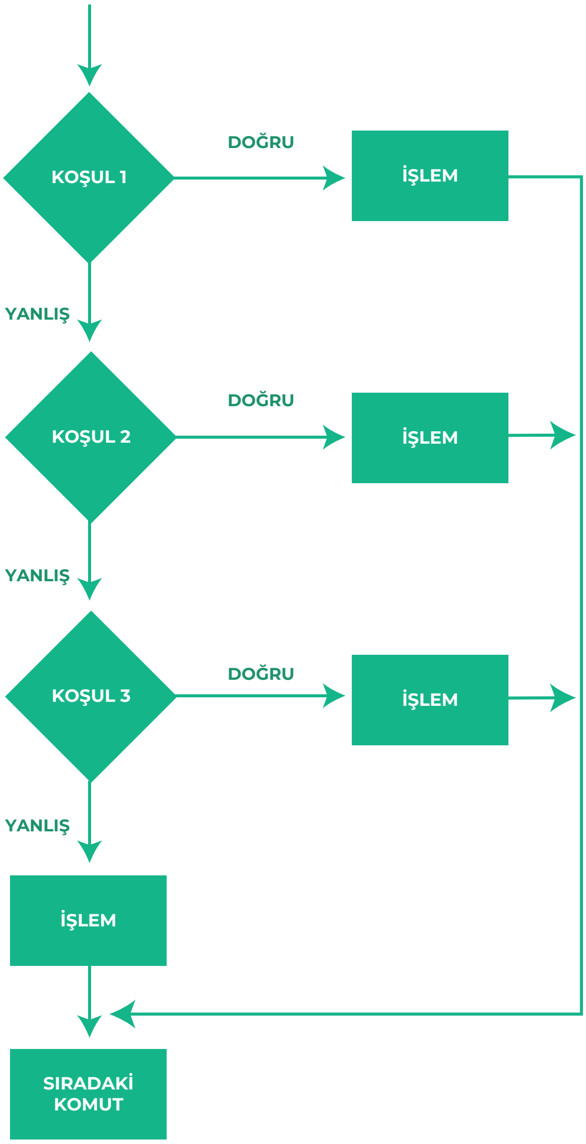
Bir programlama dilinde birden fazla koşulu kontrol etmek ve bu koşullara göre farklı kod bloklarının yürütülmesini sağlamak için kullanılan bir yapıdır. Bu yapı, koşulları sırasıyla değerlendirir ve ilk doğru koşulun kod bloğunu çalıştırır. Eğer bir koşul doğruysa diğer koşullar kontrol edilmez ve program o noktada yürütülmeye devam eder. Bu, programın akışını belirli koşullara bağlı olarak yönlendirmek için kullanılır.

Genel olarak *if-else if* yapısı şu şekildedir:

```
if (koşul1)
{
  // koşul1 doğruysa buradaki kod bloğu çalıştırılır
}
else if  (koşul2)
{
  // koşul1 yanlış, koşul2 doğruysa buradaki kod bloğu çalıştırılır
}
else
{
   // Hiçbir koşul doğru değilse buradaki kod bloğu çalıştırılır
}
```

if ve else if nedir?

- **if ifadesi:** Bir koşulun doğru olup olmadığını kontrol eder. Eğer koşul doğruysa (true), if bloğu içindeki kod çalıştırılır.
- **else if ifadesi:** İlk if koşulu doğru değilse başka bir koşulu kontrol eder. Birden fazla else if ifadesi kullanılabilir.
- **else ifadesi:** Yukarıdaki tüm koşullar yanlışsa (false), else bloğu içindeki kod çalıştırılır. else ifadesi, if ve else if ifadelerinin sonuna gelir.



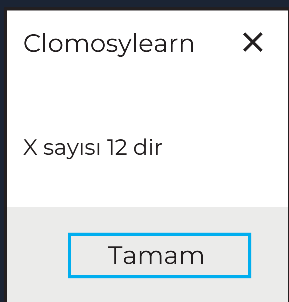
Görsel 35: If – Else If Yapısı Diyagramı

Örnek

Ana Kod

```
Var
   X : Integer;

{
   X = 12;
   if (X > 12)
      ShowMessage('X sayısı 12 den büyük') ;
   else if (X < 12)
      ShowMessage('X sayısı 12 den küçük');
   else
      ShowMessage('X sayısı 12 dir');
}
```



Kod 29: If – Else If Deyimi Örneği

İç içe if yapısı nasıl kullanılır?

Bu yapı, bir if bloğu içinde başka bir if bloğu kullanımını ifade eder. İç içe if yapısı, daha karmaşık koşulları kontrol etmek için kullanılır. Bir koşul doğru olduğunda o koşulun içinde başka bir koşulu da kontrol etmek gerektiğinde iç içe if yapıları kullanılır.

İç İçe if Yapısının Mantığı

- Bir if bloğunun içindeki kod çalıştırıldığında, bu kod içinde başka bir if bloğu olmalıdır.
- Dıştaki if koşulu doğruysa içteki if koşulu kontrol edilir.
- Bu yapı, çok katmanlı ve daha ayrıntılı koşulları kontrol etmek için kullanılır.

Bunun için basit bir örnek üzerinden kullanımına bakalım.

Bir çocuğun parkta oynamasına izin verilmesi için iki koşul var; hava güzel olmalı ve çocuğun ödevi bitmiş olmalı.

Senaryo:

- Eğer hava güzelse ve çocuğun ödevi bitmişse parka gidebilir.
- Eğer hava güzelse ama ödevi bitmemişse önce ödevini yapmalı.
- Eğer hava güzel değilse parka gidemez *(bkz.Kod 30)*.

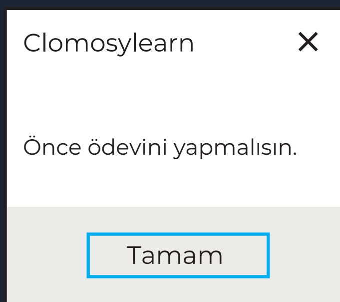
QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum5/Ornek1)

Ana Kod

```
var
  HavaGuzel, OdevBitmis: Boolean;
{
  HavaGuzel = True;  // Hava güzel
  OdevBitmis = False; // Ödev bitmemiş

  if HavaGuzel
  {
    if OdevBitmis
    {
      ShowMessage('Parka gidebilirsin.');
    }
    else
    {
      ShowMessage('Önce ödevini yapmalısın.');
    }
  }
  else
  {
      ShowMessage('Hava kötü, parka gidemezsin.');
  }
}
```



Kod 30: İç İçe If Yapısı Örneği

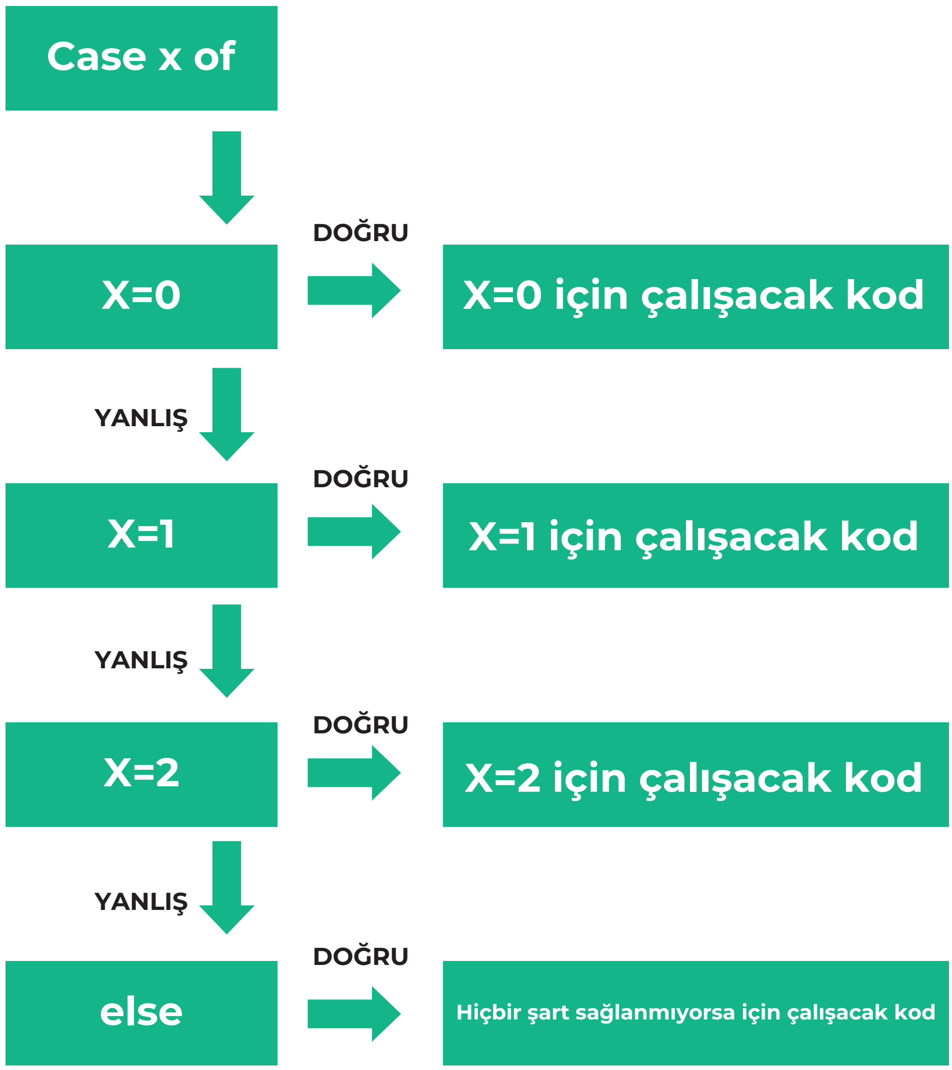
CASE (DURUM) DEYİMİ

Programlama dillerinde bulunan ve bir değişkenin farklı değerlerine göre farklı kod bloklarını çalıştıran yapıdır. Bu yapı, genellikle birden fazla koşulu kontrol etmek ve her koşula göre farklı işlemler gerçekleştirmek için kullanılır.

Case deyimi *if-else* deyimleri gibi karar kontrol mekanizmalarında kullanılır. Genellikle, karmaşık if-else bloklarına kıyasla daha okunabilir olduğu için tercih edilir. Case ile yapılabilecek karşılaştırmalar if-else ile de yapılabilir.

Genel olarak case yapısı şu şekildedir:

```
case (değişken) of
{
  değer1: // değer1 ile eşleşen durumda buradaki kod bloğu çalıştırılır
  değer2: // değer2 ile eşleşen durumda buradaki kod bloğu çalıştırılır
  değerN: // değerN ile eşleşen durumda buradaki kod bloğu çalıştırılır
else
  // Hiçbir durum eşleşmezse buradaki kod bloğu çalıştırılır
 }
```



Görsel 36: Case Yapısı Diyagramı

Örnek: Bir öğrencinin derecesini kontrol ederek hangi sırada olduğunu belirler ve buna göre bir mesaj gösterir. ‘Derece’ adında bir tam sayı (Integer) değişkeni tanımlanır ve değişkene 2 değeri atanır. Bu değişken, öğrencinin hangi sırada olduğunu temsil eder. Case yapısı kullanılarak ‘Derece’ değişkeninin değeri kontrol edilir. ‘Derece’ değişkeninin değeri 1, 2 veya 3 olduğunda, sırayla ekranda ‘1.oldu’, ‘2.oldu’ veya ‘3.oldu’ mesajları gösterilir. ‘Derece’ değişkenin değeri 1, 2 veya 3 dışında bir değerse else bloğu çalışır ve ekranda ‘Dereceye giremedi’ mesajı gösterilir. Bu adımlar, öğrencinin derecesine göre uygun mesajın ekranda görüntülenmesini sağlar *(bkz. Kod 31).*

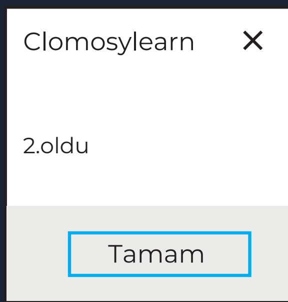
QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum5/Ornek2)

Ana Kod

```
var
   Derece:Integer;
{
   Derece=2;

  case (Derece) of
  {
    1 : ShowMessage('1.oldu');
    2: ShowMessage('2.oldu');
    3: ShowMessage('3.oldu');
  else
     ShowMessage('Dereceye giremedi');
  }
}
```



Kod 31: Case Deyimi Örneği

Alıştırmalar

1. Bir sayı belirleyip sayının pozitif bir değer olup olmadığını kontrol eden bir kod yazınız.

2. Bir sayı belirleyip sayının çift mi tek mi olduğunu kontrol eden bir kod yazınız.

3. Bir vize ve final notu belirlenip geçme notunu hesaplayınız. Hesaplanan geçme not aralığına göre bir harf döndüren (A, B, C, D, F) programı yazınız.

   Kurallar:

   - Geçme notu: (VizeNotu * 0.4) + (FinalNotu * 0.6) olarak belirleyiniz.
   - Hesaplanan not aralıklarına göre harf notları şu şekilde belirlenir:
     - 90 ve üzeri: A
     - 80 - 89: B
     - 70 - 79: C
     - 60 - 69: D
     - 60'ın altı: F

4. Bir gün numarası (1-7) alıp haftanın hangi günü olduğunu belirleyen bir kod yazınız.

5. Bir mevsim ismi (ilkbahar, yaz, sonbahar, kış) alıp bu mevsime göre bir mesaj gösteren bir kod yazınız.

6. Bir trafik ışığının durumu, belirlenen bir renk değeriyle kontrol edilecektir. Bu durumu bir if yapısı kullanarak kontrol eden bir program yazınız.

   Durumlar:

   - "Yeşil ışık" → Geçebilirsiniz.
   - "Kırmızı ışık" → Geçilemez.
   - "Sarı ışık" → Dikkatli olmalısınız.

7. Bir sisteme giriş yapan kullanıcının adı ve şifresi belirleyiniz. Eğer kullanıcı adı doğruysa şifrenin doğruluğu kontrol ediniz. Her ikisi de doğruysa "Giriş yapıldı" mesajı, yanlışsa "Hata" mesajını gösteriniz.

https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum5

Ek Kaynaklar

- If ve Case Kullanımı
  https://www.docs.clomosy.com/Conditional_Statements

BÖLÜM 6 : DÖNGÜLER VE YÖNLENDİRME İFADELERİ

Döngü deyimleri, bir programın belirli bir koşulu sağlanana kadar aynı kod bloğunu tekrar tekrar çalıştırmasına olanak tanıyan yapısal elemanlardır.

Bir süpermarket kasiyerini düşünün. Müşteriler sıraya girer ve kasiyer, her müşterinin tek tek işlemlerini yapar. Bu durum, döngülerle oldukça benzer bir mantığa sahiptir.

Örneğin; kasiyer her bir müşterinin ödemesini alır ve ardından bir sonraki müşteriye geçer. Bu, bir *for* veya *while* döngüsü ile modellenebilir. Kasada 10 müşteri olduğunu varsayalım. Kasiyer, her müşterinin işlemini tamamlayana kadar döngü devam eder. Eğer 10 müşteriden sonra yeni bir müşteri gelirse döngü tekrar başlar. Bu döngü, müşteriler bitene kadar veya çalışma saatleri bitene kadar devam eder.

Bu deyimler, programın aynı veya benzer işlemleri birçok kez gerçekleştirmesini kolaylaştırır ve tekrarlayan işlemleri otomatikleştirir.

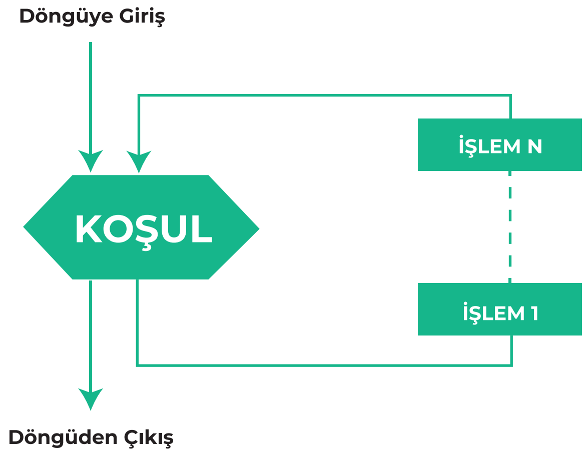
Clomosy' de üç döngü türü vardır *(bkz. Tablo 6)* :

Döngü Yapısı
While Döngüsü
Repeat – Until Döngüsü
For Döngüsü

Tablo 6: Döngü Yapısı

WHILE DÖNGÜSÜ

While döngüsü, belirli bir koşul True (doğru) olduğu sürece aynı kod bloğunu tekrar tekrar çalıştıran bir yapıdır. Döngü her adımda koşulu kontrol eder ve koşul doğru olduğu sürece çalışmaya devam eder. Eğer koşul başlangıçta False (yanlış) ise döngü hiç çalıştırılmaz. Bu döngü yapısı, bir işlemi belirli bir koşul sağlandığı sürece sürekli tekrarlamanız gerektiğinde oldukça kullanışlıdır.



Görsel 37: While Döngüsü Diyagramı

While yapısı şu şekildedir:

```
while (koşul)
{
 // Koşul doğru olduğu sürece bu
kod bloğu tekrar tekrar çalışır
}
```

Mesela, bir yazılım geliştiricisi olarak, online oyunun ödül sistemi üzerinde çalışıyorsunuz. Oyuncu, her bir seviyeyi tamamladıktan sonra yeni bir ödül kazanıyor. Ancak, oyuncunun seviyeleri bitene kadar bu ödüller otomatik olarak verilmeye devam ediyor. Her seviyeyi geçtikçe bir ödül kazanılıyor. Ödüller verilmeye devam eder, ta ki oyuncu son seviyeye gelene kadar. Son seviyeye ulaşıldığında ise ödül verilmiyor ve işlem sona eriyor.

Örnek 1

Bu örnekte, 'Benzin' adında bir tam sayı (Integer) değişkeni tanımlanır ve bu değişkene 20 değeri atanır. Bu değişken, arabanın sahip olduğu mevcut benzin miktarını temsil eder. While döngüsü başlatılır. Döngü, 'Benzin' miktarı 0'dan büyük olduğu sürece çalışmaya devam edecektir. Bu koşul, arabanın yakıtı olduğu sürece hareket edebilmesini sağlar. Her döngü adımında, "Araba gider" mesajı ekrana yazdırılır. Bu, aracın hâlen hareket ettiğini gösterir. Aynı adımda, 'Benzin' miktarından 7 birim eksiltilir. Bu, aracın her döngüde yakıt tükettiğini simüle eder. Döngü, 'Benzin' miktarı 0 veya daha az olduğunda sona erer. Bu durumda, "Yakıt bitti. Araba gidemez." mesajı ekrana yazdırılır*(bkz. Kod 32).*

Bu adımlar, arabanın belirli bir miktar yakıtla kaç kez hareket edebileceğini simüle eder. Döngü, yakıt tükendiğinde sona erer ve araba durur.

QR Okut

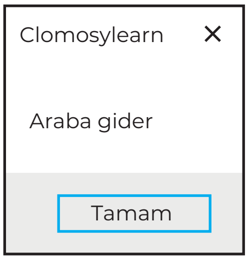
(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum6/Ornek1)

Ana Kod

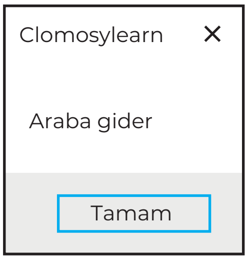
```
var
  Benzin:Integer;
{
  Benzin = 20;
   while (Benzin > 0)
   {
      ShowMessage('Araba gider');
      Benzin = Benzin - 7;
   }
   ShowMessage('Yakıt bitti araba gidemez');
}
```

Kod 32: While Döngüsü Örneği 1

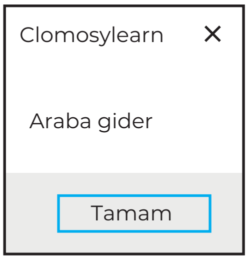
Kod, 3 defa döngüye girer ve mesajı verir *(bkz. Görsel 38, 39, 40)*. Ardından döngü, koşulu sağlamadığı için sonlanır ve döngü çıkışındaki mesajı gösterir *(bkz. Görsel 41).*



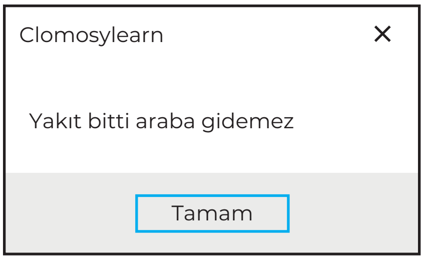
Görsel 38: While Döngü Sonucu 1



Görsel 39: While Döngü Sonucu 2



Görsel 40: While Döngü Sonucu 3



Görsel 41: While Döngü Sonu Mesajı

Örnek 2

‘KalanPara’ adında bir tam sayı değişkeni tanımlanır ve başlangıç değeri olarak 200 atanır. Bu değişken, babanın cebinde kalan parayı temsil eder. Bir while döngüsü başlatılır. Döngü, ‘KalanPara’ 0'dan büyük olduğu sürece çalışacaktır. Yani, babanın cebinde para olduğu sürece işlem devam eder. Her döngü adımında, ‘KalanPara’ değerinden 50 lira çıkarılır. Bu işlem, babanın oğluna her seferinde 50 lira verdiğini simgeler. Ekranda önce “Baba para ver” mesajı gösterilir, ardından ‘KalanPara’ değerini içeren “Al oğlum 50 lira, cebimde *X* liram kaldı” mesajı görüntülenir. *X*, kalan para miktarını temsil eder. ‘KalanPara’ 0 veya daha düşük olduğunda, döngü sona erer ve ekrana “Para bitti” mesajı yazdırılır. Bu süreç, babanın oğluna belirli bir miktar para verdikçe cebindeki paranın azaldığını simüle eder. Cebindeki para bittiğinde döngü sona erer ve işlem tamamlanır *(bkz. Kod 33).*

QR Okut

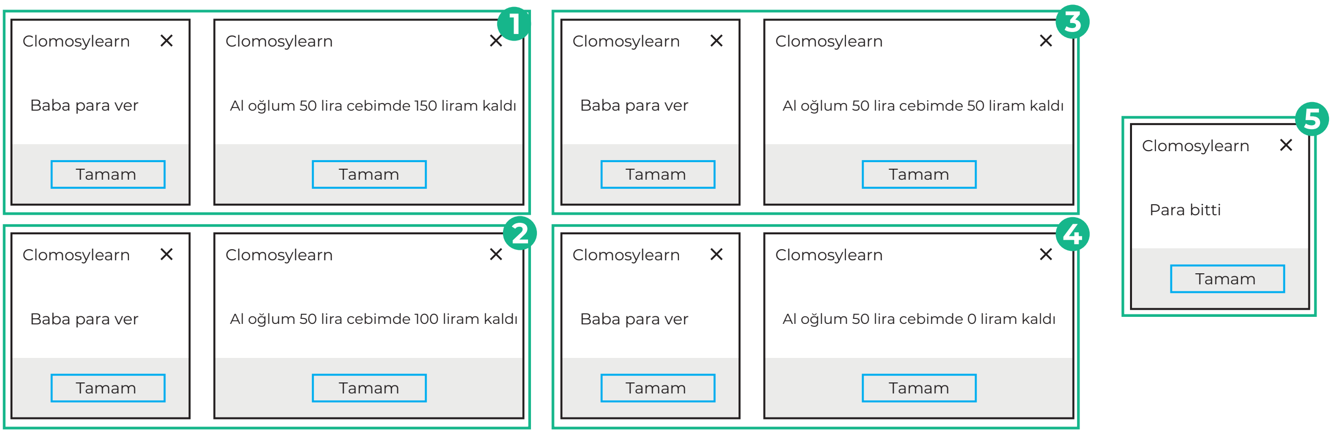
(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum6/Ornek2)

Ana Kod

```
var
  KalanPara : Integer ;
{
  KalanPara = 200;
  while (KalanPara > 0)
  {
     KalanPara = KalanPara - 50;
     ShowMessage('Baba para ver');
     ShowMessage('Al oğlum 50 lira cebimde '+IntToStr(KalanPara)+' '+' liram kaldı');
  }
    ShowMessage('Para bitti');
}
```

Kod 33: While Döngüsü Örneği 2

Sonuç olarak döngü 4 kez çalışır ve ardından “Para bitti” sonucunu gösterir.



Görsel 42: Örnek 2 Sonucu

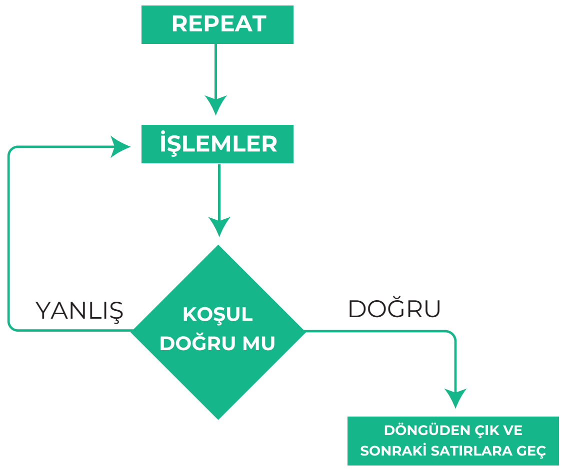
REPEAT-UNTIL DÖNGÜSÜ

Repeat-until döngüsü, belirli bir koşul sağlanana kadar belirli bir kod bloğunu tekrar tekrar çalıştıran bir döngüdür.

Bu döngü türü, koşulun döngü içinde değil, döngünün sonunda kontrol edildiği bir yapıya sahiptir. Bu nedenle, repeat-until döngüsü, döngü bloğunun **en az bir kez** çalışmasını garanti eder.

Repeat-until döngüsü şu şekildedir:

```
repeat
{
  // Bu kod bloğu en az bir kez çalışır
}
until (koşul); //koşul False olduğunda
devam eder
```



Görsel 43: Repeat – Until Döngüsü Diyagramı

Mesela, bir uzay keşif aracının Mars'a iniş yaparken, atmosferin belirli koşullarına uygun bir hızda iniş yapabilmesi için sürekli olarak iniş hızını ayarlaması gerekir. İnişe başladığında araç ilk olarak hızını kontrol eder ve atmosferin etkilerini izlemeye başlar. Eğer hız çok fazla ise frenleme sistemi devreye girer ve iniş hızı azaltılır. Bu işlem, hız güvenli bir seviyeye ulaşana kadar devam eder. Ancak, iniş hızı güvenli bir seviyeye ulaştığında frenleme sistemi durur ve iniş tamamlanır.

Bu süreç, repeat-until döngüsüyle yapılabilir. Araç, ilk hız kontrolünü yaptıktan sonra güvenli iniş hızına ulaşana kadar hızını kontrol etmeye devam eder. Hız güvenli seviyeye geldiğinde döngü sona erer ve iniş tamamlanır.

Örnek

Bu örnekte, ‘Sayi’ adlı bir tam sayı değişkeni tanımlanır ve başlangıç değeri 0 olarak atanır. Repeat...until döngüsü, belirli bir koşul sağlanana kadar en az bir kez çalışacak şekilde ayarlanmıştır. Döngü her çalıştığında ‘Sayi’ değeri 4 birim artırılır. Her artırma işleminden sonra güncel ‘Sayi’ değeri ekrana “Sayı: *X*” mesajıyla yazdırılır (*X*, o anki ‘Sayi’ değeridir.). Döngü, koşul sağlanana kadar devam eder. Bu durumda, ‘Sayi’ değişkeni 3 kez artırılır. Her döngüde ‘Sayi’ sırasıyla 4, 8 ve 12 olur. Üçüncü artırma işleminden sonra ‘Sayi’ değeri 12 olur ve koşul bir sonraki adımda doğru hale gelir. Bu noktada döngü sonlanır. Bu yapı, döngünün en az bir kez çalışmasını ve belirli bir koşul sağlanana kadar tekrar etmesini sağlar *(bkz. Kod 34)*.

QR Okut

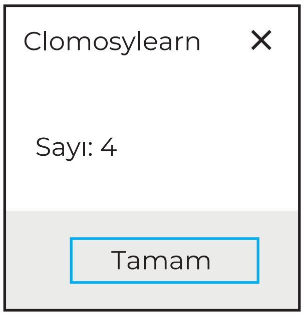
(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum6/Ornek3)

Ana Kod

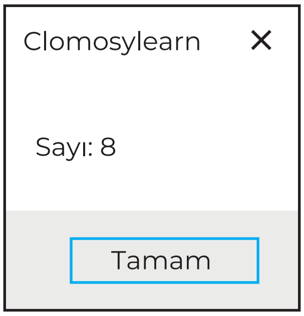
```
var
  Sayi: Integer;
{
  Sayi = 0;
  repeat
    Sayi = Sayi + 4;
    ShowMessage('Sayı: ' + IntToStr(Sayi));
  until (Sayi > 10);
    ShowMessage('Sayı 10 dan büyük olduğu için döngüden çıkıldı.');
}
```

Kod 34: Repeat – Until Döngüsü Örneği

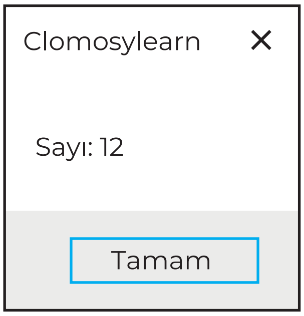
DİKKAT!
Repeat-Until döngüsünde koşul False olduğunda döngüye devam eder. True olduğunda döngüyü sonlandırmaktadır.



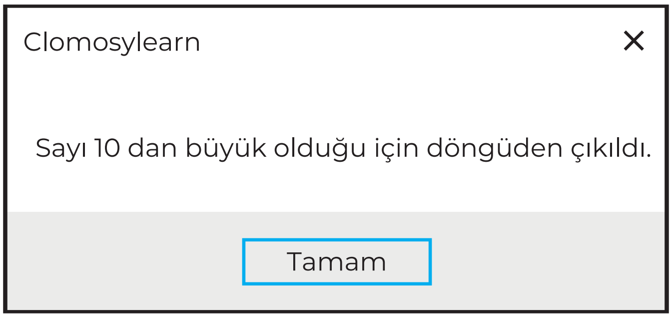
Görsel 44: Repeat-Until Döngü Sonucu 1



Görsel 45: Repeat-Until Döngü Sonucu 2



Görsel 46: Repeat-Until Döngü Sonucu 3



Görsel 47: Repeat-Until Döngü Sonucu Mesajı

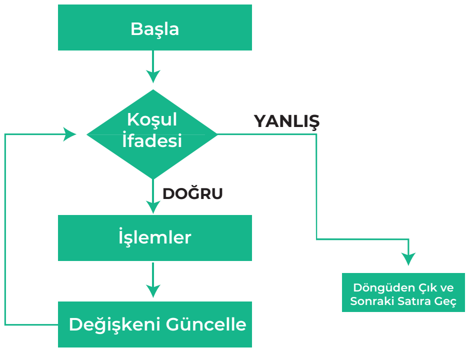
FOR DÖNGÜSÜ

For döngüsü, belirli bir başlangıç noktasından belirli bir son noktaya kadar bir kod bloğunu tekrar tekrar çalıştıran bir döngü yapısıdır. While ve Repeat-Until döngüleri gibi bir kod parçasını belirli koşullar altında tekrar etmek için kullanılır.

Bu döngü, bir döngü değişkeni kullanılarak döngünün her adımında değişen bir kontrol mekanizması sağlar.

For döngüsü şu şekildedir:

```
for ( Döngü sayacı = Başlangıç değeri to Son Değer )
{
   // Belirtilen koşul sağlandığı sürece bu kod bloğu tekrar tekrar çalışır
}
```



Görsel 48: For Döngüsü Diyagramı

Mesela, bir müzik uygulaması geliştiriliyor ve kullanıcılar, çalma listelerine belirli sayıda şarkı ekleyebiliyorlar. Her kullanıcı çalma listesine şarkı ekledikçe eklenen şarkılar sırayla çalmaya başlıyor. Bu durumda uygulama, çalma listesindeki tüm şarkıları sırasıyla çalmak için bir döngü kullanılır.

Bu işlem, for döngüsü kullanılarak gerçekleştirilir. For döngüsü, çalma listesindeki her şarkıyı sırayla çalmak için uygun olandır.

Örnek

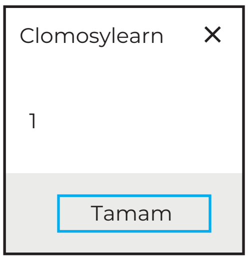
1'den 3'e kadar olan doğal sayıları yazdırmak için bir kod oluşturulur. Bu işlem, bir döngü kullanılarak gerçekleştirilebilir. Bu işlem for döngüsü kullanılarak şu şekilde yapılabilir:

Ana Kod

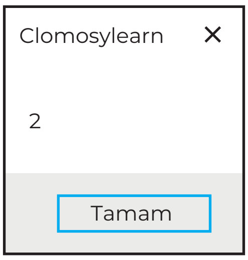
```
var
  Sayi:Integer;
{
  for (Sayi=1 to 3)
  {
    ShowMessage(Sayi);
  }
}
```

Kod 35: For Döngüsü Örneği

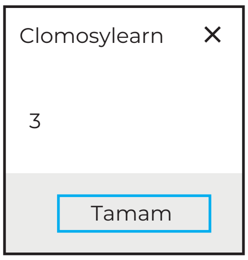
Kod 35'te yer alan kodun çalıştırılması ile alınan çıktı *Görsel 49, 50, 51*'deki gibidir:



Görsel 49: To Kullanımı Sonucu 1



Görsel 50: To Kullanımı Sonucu 2



Görsel 51: To Kullanımı Sonucu 3

Bu örnekte for döngüsünde, "Sayi" değişkenine 1 değeri ilk olarak atandı. 3 değerine ulaşana kadar her adımda bu değişken 1 arttırıldı. Her artıştan sonra ShowMessage fonksiyonu kullanılarak "Sayi" değişkeninin mevcut değeri ekrana yazdırıldı. *to* anahtar kelimesi, döngünün bitiş değerini belirledi. Bu durumda, döngü toplamda 3 kez çalıştı.

Döngü sona erdiğinde, süslü parantezler ({...}) arasındaki tüm komutlar çalıştırılmış olur ve program akışı döngü sonrasındaki kodlarla devam eder.

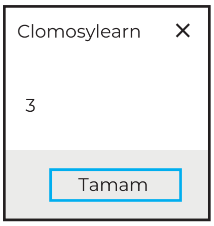
BİLGİLENDİRME:
For döngüsünde **to** anahtar kelimesi ile sayılar belirlenen değere kadar arttırılarak döngü devam ederken **downto** anahtar sözcüğü ile sayılar azaltılarak döngü ilk değere ulaşana kadar devam eder.

Aynı örnek bu sefer 3'ten 1'e kadar azalarak gitsin. Bunun için *downto* kullanılır *(bkz. Kod 36).*

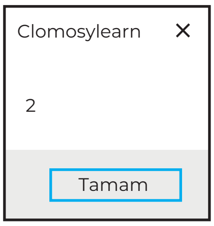
Ana Kod

```
var
   Sayi:Integer;
{
  for (Sayi=3 downto 1)
  {
     ShowMessage(Sayi);
  }
}
```

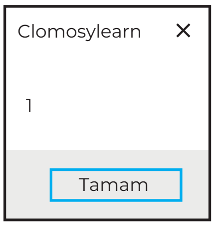
Kod 36: Downto Kullanım Örneği



Görsel 52:Downto Kullanımı Sonucu 1



Görsel 53: Downto Kullanımı Sonucu 2



Görsel 53: Downto Kullanımı Sonucu 3

Döngülerin nasıl kullanıldığını öğrendik. Şimdi, bu döngüler arasındaki farkları açıklayarak hangi durumlarda kullanılabileceklerine bakalım:

- **For Döngüsü:**

For döngüsü, belirli bir sayıda tekrarlanması istenilen işlemler için kullanılır. Döngü, bir sayaç değişkenini, başlangıç değerinden bitiş değerine kadar kontrol eder. Genellikle tekrar sayısının önceden bilindiği durumlarda tercih edilir.

Örnek

Bir öğretmen, sınav sonuçlarını dağıtırken, sınıftaki her öğrenciye sırayla notlarını verir. Sınıfta kaç öğrenci olduğunu bildiği için bu işlem, son öğrenciye not verildiğinde tamamlanır.

- **While Döngüsü:**

While döngüsü, belirli bir koşul sağlandığı sürece çalışmaya devam eder. Bu döngüde, koşul döngünün başında kontrol edilir ve eğer koşul sağlanmıyorsa döngü hiç çalışmayabilir. Genellikle tekrar sayısının önceden bilinmediği, dinamik durumlar için kullanılır.

Örnek

Bir sürücü, aracının deposunu doldurmak için benzin istasyonuna gider. Sürücü, depo tamamen dolana kadar yakıt doldurmaya devam eder. Depo dolduğunda ise doldurma işlemi sona erer.

- **Repeat - Until Döngüsü:**

Repeat-until döngüsü, while döngüsüne benzer ancak en az bir kez çalışması istenir. Döngüde önce işlem yapılır, ardından koşul kontrol edilir. Eğer koşul sağlanmıyorsa döngü devam eder. Bu, özellikle bir işlemin en az bir kez yapılması gerektiğinde kullanılır.

Örnek

Bir aşçı, yemeği hazırlarken her seferinde tadına bakar. Eğer yemeğin tuzu eksikse tuz ekleyip tekrar tadına bakar. Bu işlem, yemeğin tadı istenilen seviyeye gelene kadar devam eder.

İç İçe For Döngüsü

Bir for döngüsünün içinde başka bir for döngüsünün yer aldığı bir yapıdır.

Bu yapı, genellikle çok boyutlu verilerle çalışırken veya karmaşık tekrarlayan işlemler yaparken kullanılır. İç içe for döngüleri, iki veya daha fazla boyutlu diziler üzerinde işlem yapmak, tablo oluşturmak veya birbirine bağlı işlemleri gerçekleştirmek için kullanılır.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum6/Ornek4)

Ana Kod

```
var
 Satir,Sutun:Integer;
{
 for Satir = 1 to 5 //dış döngü
 {
   for Sutun = 1 to 5 //iç döngü
   {
      // Satır ve Sütun'a bağlı işlemler
      Console.Text(IntToStr(Satir)+'-'+IntToStr(Sutun));
   }
 }
}
```

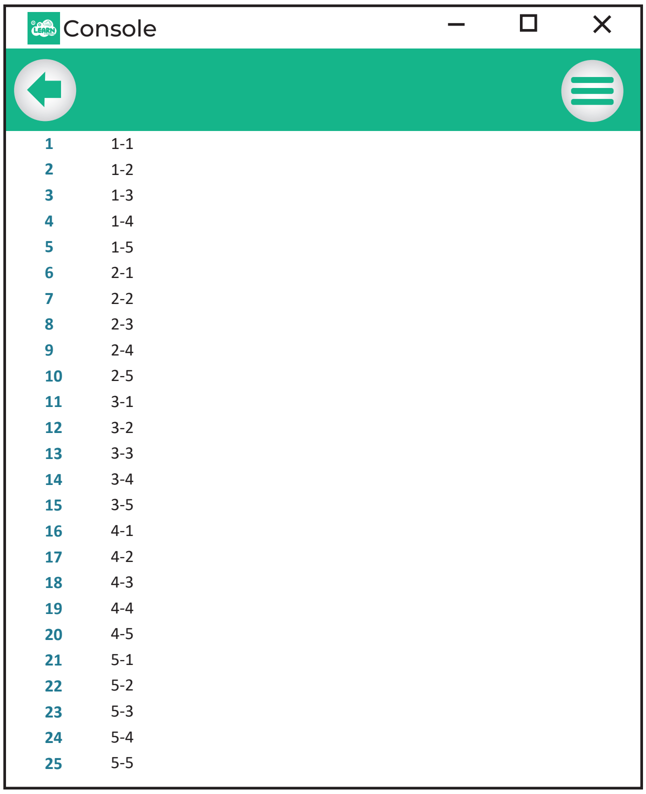
Kod 37: İç İçe For Döngüsü Örneği

İç içe for döngüsü, bir dış döngü ve onun içinde bir iç döngüden oluşur. Döngü içindeki her şey belirtilen sayıda tekrarlanır.

Kod 37'de, dış döngü 5 kez tekrarlanır. Dış döngü her tekrarlandığında, iç döngü de baştan sona çalışır. Eğer dış döngü 5 kez ve iç döngü de 5 kez tekrarlanırsa iç döngü her seferinde 5 kez çalışacağı için toplamda 25 kez işlem gerçekleşir. Bu durum tekrar sayılarının çarpımı, 5 * 5'in 25'e eşit olmasıyla açıklanabilir.

Kod 37'de verilen for döngüsü çalıştırıldığında konsolda nasıl bir çıktı alınır?

Denemek için kod, geliştirici ortamına yazılır. Ve Windows üzerinde çalıştırıldığında *Görsel 55*'deki gibi sonuç elde edilir.



Görsel 55: İç İçe For Döngüsü Konsol Sonucu

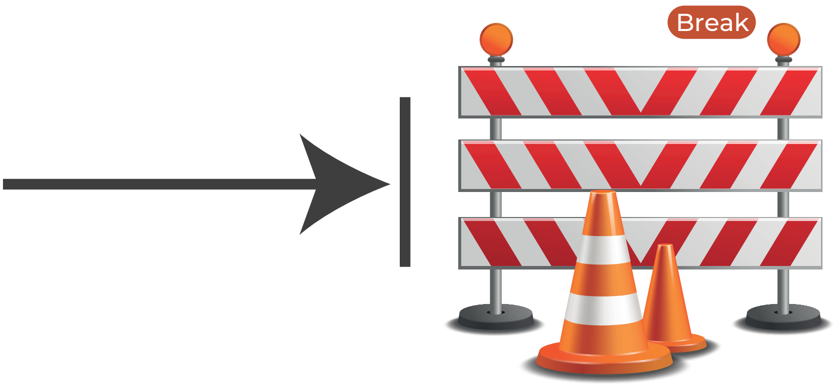
DÖNGÜ YÖNLENDİRME İFADELERİ

Döngülerin normal akışını değiştirmek için kullanılan ifadeleri temsil eder. Bu ifadeler, döngülerin belirli koşullar altında belirli bir adıma atlamasını, döngüyü tamamen sonlandırmasını veya bir sonraki döngü adımına geçmesini sağlar.

Döngü yönlendirme ifadeleri şunlardır:

BREAK

Break komutu, döngülerin çalışmasını anında durduran ve döngüden çıkmasını sağlayan bir komuttur. Program döngünün hemen ardından çalışmaya devam eder. Bu komut, döngünün ortasında belirli bir koşul sağlandığında döngüyü sonlandırmak için kullanılır.



Genel yapısı:

```
for (i = 1 to 10)
{
        if (koşul)
          break; // Döngüyü sonlandırır
      // Diğer işlemler
}
```

Örneğin: 1'den 10'a kadar döngü çalışmaktadır. Değer 3'e geldiğinde program tarafından break komutu çalıştırılır ve döngü sonlandırılır. Döngü sona erdikten sonra program "Döngü sonlandı." mesajı gösterir *(bkz. Kod 38).*

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum6/Ornek5)

Ana Kod

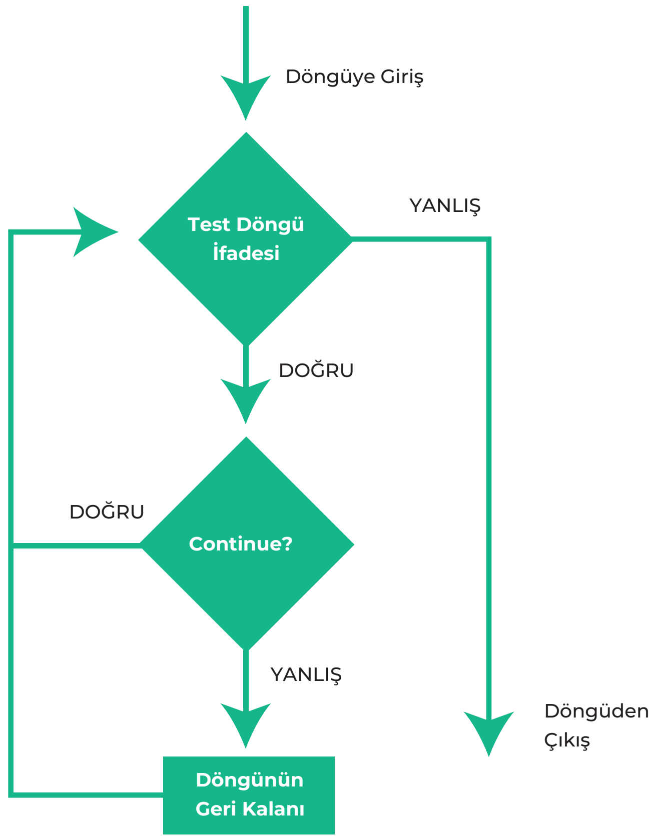
```
var
   i : Integer;
{
   for(i = 1 to 10)
  {
      ShowMessage('Değer : '+ IntToStr(i));
      // i değeri 3 olduğunda döngüden çık
   if(i == 3)
   {
       ShowMessage('i değeri 3 olduğu için döngüden çıkıyorum.') ;
       Break;
   }
  }
    ShowMessage('Döngü sonlandı.');
}
```

Kod 38: Break Komutu Örneği

CONTINUE

Döngünün mevcut adımını atlayarak bir sonraki adıma geçmesini sağlayan bir ifadedir. Döngü içinde belirli bir koşul sağlandığında, döngünün o anki yinelemesini atlayıp bir sonraki yinelemeye geçmesi sağlanır. Bu, belirli işlemleri atlamak veya bir koşul gerçekleştiğinde döngünün komuttan sonra geri kalan işlemlerini yapmamak için kullanılır.





Görsel 56: Continue Diyagramı

Kod 39'da döngü 1'den 4'e kadar döner. 2'ye geldiğinde 2 değerini ekrana yazdırmaz ve döngünün kalan kod satırlarını çalıştırmaya devam eder.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum6/Ornek6)

Ana Kod

```
var
   i: Integer;
{
   for (i = 1 to 4)
   {
      // i değeri 2 olduğunda bu iterasyonu atla, diğer iterasyona geç
   if (i == 2)
   {
        ShowMessage('i değeri 2 olduğu için bu tekrar ifadesini atlıyorum.');
        Continue;
        //Continue komutundan sonra gelen durumları atlar
   }
     // 2 değeri haricindeki diğer i değerlerini ekrana yazdırır
     ShowMessage('Döngüdeyim, i = '+ IntToStr(i));
  }
}
```

Kod 39: Continue Komutu Örneği

EXIT

Döngü kontrol ifadesi olmamakla birlikte bu ifade, döngü içinde kullanıldığında hem döngüyü sonlandırır hem de bulunduğu yerel (local) metottan çıkış yapar. Break ifadesinden farklı olarak, döngüden sonra yerel metottaki kod satırları da çalıştırılamaz.



QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum6/Ornek7)

Ana Kod

```
var
   i: Integer;
{
   for (i = 1 to 10)
   {
      ShowMessage('Değer : '+ IntToStr(i));
   if (i == 5)
   {
      ShowMessage('i değeri 5 olduğu için döngüden çıkıyorum.');
      Exit;
   }
  }
      ShowMessage('Döngü sonlandı.');
}
```

Kod 40: Exit Komutu Örneği

Alıştırmalar

1. Bir sayı belirleyip bu sayıdan geriye doğru sayarak 1'e kadar olan sayıları ekranda gösteren bir kod yazınız.

2. 1'den 10'a kadar olan sayıların karesini hesaplayıp ekranda gösteren bir kod yazınız.

3. Bir sayı belirleyip bu sayının faktöriyelini hesaplayarak ekranda gösteren bir kod yazınız.

4. 1'den 10'a kadar olan sayılardan sadece çift olanları ekranda gösteren bir program yazınız. Tek sayıları göz ardı ediniz.

5. 0'dan 10'a kadar dönen bir döngüde 5'ten büyük ilk sayı bulunduğunda, işlemi sonlandıran programı yazınız.

https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum6

Ek Kaynaklar

- While Kullanımı
  https://www.docs.clomosy.com/Loops#While_Loop

- Repeat-until Kullanımı
  https://www.docs.clomosy.com/Loops#Repeat_-_Until_Loop

- For Kullanımı
  https://www.docs.clomosy.com/Loops#For_Loop

BÖLÜM 7 : DİZİLER

Futbol Millî Takımı 2024 Avrupa Futbol Şampiyonası çeyrek final maçında sahaya çıkan oyuncuların bir listesi oluşturuldu. Bu liste için her bir oyuncu string türünde bir değişkene atandı.

```
futbolcu1 = 'Mert';
futbolcu2 = 'Ferdi';
futbolcu3 = 'Abdülkerim';
futbolcu4 = 'Samet';
futbolcu5 = 'Kaan';
futbolcu6 = 'Mert';
futbolcu7 = 'Kenan';
futbolcu8 = 'Hakan';
futbolcu9 = 'Salih';
futbolcu10 = 'Arda';
futbolcu11 = 'Barış';
```



Millî Takım kadrosunu saklamak için string türünde 11 farklı değişken oluşturuldu. Oysa tüm kadro tek bir değişkende tutulabilirdi. Bunu yaparken yine her oyuncuya tek tek ulaşılabilir. Peki, nasıl yapılır? Dizilerle.

Diziler, aynı türdeki verileri bir araya getirip sıralı bir şekilde saklamak için kullanılan veri yapılarıdır. Yani burada 11 tane string türünde değeri saklayabilen bir dizi oluşturup futbolcular bu dizinin elemanları olarak atanabilir. Dizi burada 11 çekmeceli bir dolap gibi davranır ve her bir çekmecesinde farklı bir veriyi saklayabilir.

Hadi yapalım. Öncelikle dolabımızı yani dizimizi oluşturalım.

```
MilliTakim: array[10] of String;
```

Dizi ismi 'milliTakim' olarak adlandırıldı. *Array* ifadesi, bir dizi oluşturulduğunu belirtmek için kullanılır ve yanındaki köşeli parantezin içine, dizinin eleman sayısını belirlemek için bir tam sayı yazılır. String ifadesi ise dizinin elemanlarının String türünde yani metinsel olacağını belirtir. Dizinin elemanlarının aynı türde olduğu unutulmamalıdır!

Bir şey dikkatini çekti mi? Millî Takım 11 kişi fakat parantez içine 10 yazıldı. Çünkü diziler saymaya 1'den değil 0'dan başlar. Dizinin elemanlarının konumları *indis* olarak adlandırılır. Yani dizinin ilk elemanı 0. indise, sonraki eleman 1. indise yerleştirilir. Bu şekilde devam edildiğinde 0. indisten 10. indise kadar takımın 11 oyuncusu diziye yerleştirilir.

Dizinin her bir elemanına, dizinin adı ve elemanın indisi kullanılarak erişilebilir. Örneğin; dizi[0] ifadesi dizinin ilk elemanını, dizi[1] ifadesi ikinci elemanını temsil eder.

```
MilliTakim[0]='Mert';
```

Artık dizinin 0. indisinin tuttuğu değer 'Mert'. Diğer futbolcuların, dizinin elemanları olarak indis değerlerine göre bu şekilde ataması yapılır. Ya da aşağıdaki yöntem kullanılarak bütün futbolcular diziye tek seferde atanabilir.

```
MilliTakim = ['Mert', 'Ferdi', 'Abdülkerim', 'Samet', 'Kaan', 'Mert', 'Kenan',
'Hakan', 'Salih', 'Arda', 'Barış'];
```

Aşağıda, indisi 1 olan futbolcunun yani Ferdi'nin ismi ekrana yazdırıldı.

```
ShowMessage(MilliTakim[1]); // Ferdi
```

Peki, köşeli parantezin içine 10'dan büyük bir sayı yazmayı denedin mi?

Eğer indis değerine dizide bulunmayan yani dizinin sınırlarını aşan bir değer yazılırsa hata alınır. Bu yüzden dizinin sınırlarını bilmek önemlidir. İngilizce'de uzunluk anlamına gelen *Length* kelimesinin yanına parantez içinde, dizinin adı yazılarak dizinin kaç eleman içerdiği öğrenilir. Örneğin aşağıdaki ifade 11 değerini verir.

```
ShowMessage(Length(MilliTakim)) //11
```

Peki, Millî Takımın 11 oyuncusu sırayla ekrana yazdırılmak istendiğinde; 11 tane ShowMessage mı yoksa aynı işlemi tekrar tekrar yapabilen yapılar mı kullanılmalıdır.

Döngüler! Aynı işlemi tekrar tekrar gerçekleştiren yapılardır. Döngü yapıları, diziler üzerinde çeşitli işlemler yaparken büyük kolaylık sağlar.

Buraya kadar anlatılanları birleştirerek bir örnek oluşturuldu. Dizi tanımlandı, elemanları atandı ve futbolcuların isimleri bir for döngüsü içinde ekrana yazdırıldı *(bkz. Kod 41)*.

Ana Kod

```
var
  MilliTakim: array[10] of String;
  i:integer;
{
  MilliTakim = ['Mert', 'Ferdi', 'Abdülkerim', 'Samet', 'Kaan', 'Mert', 'Kenan', 'Hakan', 'Salih', 'Arda', 'Barış'];

  for(i=0 to Length(MilliTakim)-1)
  {
    ShowMessage(MilliTakim[i]);
  }
}
```

Kod 41: Diziler Örneği

Kod 41'de -1 fark edildi mi? Length ifadesi kullanılarak dizinin eleman sayısı öğrenilebiliyordu. Dizinin son indisi ise her zaman bu değerin bir sayı eksiğidir. Bu yüzden döngü "for(i=0 to Length(MilliTakim)-1)" şeklinde oluşturulduğunda döngü dizinin 0.indisinden son indisine kadar bütün dizi elemanları üzerinde döner.

Kod 42'de, 'Abdülkerim' oyundan çıkarılıp yerine 'Merih' alındı. Yani dizinin 2. indisinde tutulan değer 'Merih' olarak değiştirildi.

```
MilliTakim[2] = 'Merih';
```

İşte bu kadar kolay. Artık dizinin ikinci indisinde tutulan değer 'Merih' oldu. *Kod 42* çalıştırıldığında "MilliTakim[2]='Merih';" ifadesinden sonra 2.indisteki değerin nasıl değiştiği görülebilir.

Ana Kod

```
var
  MilliTakim: array[10] of String;
  i:integer;
{
  MilliTakim = ['Mert', 'Ferdi', 'Abdülkerim', 'Samet', 'Kaan', 'Mert', 'Kenan', 'Hakan', 'Salih', 'Arda', 'Barış'];

    ShowMessage(MilliTakim[2]);

  MilliTakim[2] = 'Merih';
    ShowMessage(MilliTakim[2]);
}
```

Kod 42: Dizi Elemanları Değişimi 1

'Barış' ve 'Arda'nın yerleri değiştirilmek istendiğinde dizinin 9.indisine 'Arda', 10.indisine 'Barış' yerleştirilir. Bunun için 'Tut' adı verilen string türünde yeni bir değişken tanımlanır ve dizinin 9.indisi 'Tut' değişkenine atanır *(bkz. Kod 41).*

```
Tut = MilliTakim[9];
```

'Tut' değişkeni artık 'Arda' değerini aldı.
Bu sefer dizinin 10.indisindeki değer 9.indise atandı.

```
MilliTakim[9] = MilliTakim[10];
```

Artık hem 9 hem de 10.indis 'Barış' değerini tutuyor. Son olarak 10.indise 'Tut' değişkeni atanarak 10.indis de 'Arda' olarak güncellenmiş olsun.

```
MilliTakim[10]=tut;
```

'Arda' ve 'Barış'ın yerleri değiştirildi. Artık bir for döngüsüyle dizi ekrana yazdırılabilir.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum7/Ornek1)

Ana Kod

```
var
  MilliTakim: array[10] of String;
  i:integer;
  Tut:String;

{
  MilliTakim = ['Mert', 'Ferdi', 'Abdülkerim', 'Samet', 'Kaan', 'Mert', 'Kenan', 'Hakan', 'Salih', 'Arda', 'Barış'];

 for(i=0 to Length(MilliTakim)-1)
 {
   ShowMessage(MilliTakim[i]);
 }

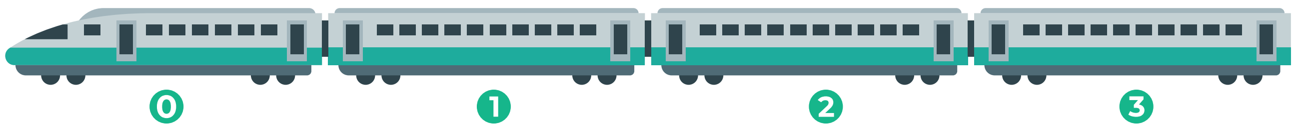
  Tut = MilliTakim[9];
  MilliTakim[9] = MilliTakim[10];
  MilliTakim[10] = tut;

  ShowMessage('Yer değiştirildi.');
 for(i=0 to Length(MilliTakim)-1)
 {
   ShowMessage(MilliTakim[i]);
 }
}
```

Kod 43: Dizi Elemanları Değişimi 2

Artık dizilerle tanışıldı ve ilk dizi tanımlaması yapıldı. Dizi üzerinde bazı örnek kodlamalar yapıldı. Programlama dillerinde sıklıkla kullanılan diziler, veri yönetimi ve işlemeyi daha kolay hale getirir. Dizilerin en önemli özelliklerinden biri, elemanlarının sıralı ve ardışık bir şekilde saklanmasıdır. Bu sıralama, günlük hayattan farklı olarak sıfırdan başlar. Yani, dizinin ilk elemanı sıfırıncı indise, sonraki eleman birinci indise ve devam eden elemanlar da sırasıyla artan indis değerlerine atanır.

Diziler, çeşitli türlerde veri saklamak için kullanılabilir; tam sayılar, metinsel ifadeler veya ondalık sayılar gibi. Diziler üzerinde çeşitli işlemler yapılabilir. Bu işlemler arasında eleman ekleme, çıkarma, elemanları sıralama, belirli bir elemanı arama ve diziyi birleştirme gibi işlemler yer alır. Ayrıca, döngüler kullanılarak diziler üzerinde toplu işlemler gerçekleştirmek mümkündür. Bir önceki örnekte futbolcu isimlerini yazdırırken yapıldığı gibi bir for döngüsü kullanarak dizideki tüm elemanlar sıralı bir şekilde işlenebilir.



Diziler tren vagonlarına benzetilebilir. Bir tren, birbirine bağlı ve belirli bir sırayla dizilmiş vagonlardan oluşur. Her vagon, trenin bir parçasıdır. Tıpkı tren vagonlarının ardışık ve sıralı bir şekilde dizilmesi gibi diziler de elemanlarını belirli bir sırayla saklar. Tren vagonları, belirli bir düzene göre dizilmiştir ve bu düzen yolculuğun başından sonuna kadar korunur. Dizilerin de elemanlarını saklama sırası dizinin oluşturulmasından itibaren korunur.

Aynı şekilde, dizideki her eleman da belirli bir indise sahiptir. Bir trende belirli bir vagona erişmek için vagonun numarasını bilmek yeterlidir; dizilerde de aynı mantık geçerlidir ve belirli bir elemanı bulmak için elemanın indis numarasını bilmek yeterlidir. Tren vagonlarının her biri belirli bir türde yük veya yolcu taşımak için tasarlanmış olabilir. Aynı şekilde, diziler de aynı türdeki verileri

Daha önce bir dizi tanımlaması yapıldı fakat Clomosy'de dizi tanımlamanın tek yolu bu değil. *Array* ifadesinin yanına parantez içinde başlangıç ve bitiş indisini vererek de dizi tanımlanabilir.

```
var
        DiziAdi: array[başlangıçIndisi..bitişIndisi] of  VeriTürü;
        DiziAdi: array[dizi_boyutu] of VeriTürü;
```

Her iki yol ile birer dizi tanımlaması aşağıdaki gibidir:

```
var
        Sayilar: array[0..4] of Integer;        // 5 tam sayıyı içeren bir dizi
        Sayilar2: array[2] of Integer;          // 3 tam sayıyı içeren bir dizi
```

Oluşan dizileri inceleyelim:

Sayilar dizisi

Sayilar[0]	Sayilar[1]	Sayilar[2]	Sayilar[3]	Sayilar[4]

Sayilar2 dizisi

Sayilar2[0]	Sayilar2[1]	Sayilar2[2]

Dizinin hangi türde kaç eleman içereceğini belirterek dizi oluşturuldu fakat henüz dizinin elemanlarına değer atanmadı. Yani trenin vagonları henüz boş. Şimdi 'Sayilar2' dizisinin elemanlarına indis değerlerini kullanarak erişip değer ataması yapalım.

```
Sayilar2[0]=10;
Sayilar2[1]=34;
Sayilar2[2]=42;
```

Değer atamalarından sonra 'Sayilar2' dizisi:

Sayilar2[0]	Sayilar2[1]	Sayilar2[2]
10	34	42

Örnek 1

Kod 44'te eleman olarak yılın aylarını içeren 'Dizi' isimli bir string dizisi tanımlandı ve dizinin elemanları ekrana yazdırıldı.

Ana Kod

```
var
   Dizi: array of String;
   i:integer;
{
   Dizi = ['Ocak', 'Şubat', 'Mart', 'Nisan', 'Mayıs', 'Haziran', 'Temmuz', 'Ağustos', 'Eylül', 'Ekim', 'Kasım', 'Aralık'];
   for (i = 0 to Length(Dizi) - 1)
        ShowMessage('Dizi['+ IntToStr(i) + '] = ' + Dizi[i]);
}
```

Kod 44: Dizi Örneği 1

Örnek 2

Verilen bir dizi içerisine, rastgele sayılar yerleştirildi ve bu dizi konsola yazdırıldı. (Dizi elemanlarını 0 ile 30 arasında rastgele sayılarla dolduruldu ve dizi boyutu 5 olarak belirlendi.)*(bkz. Kod 45)*.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum7/Ornek2)

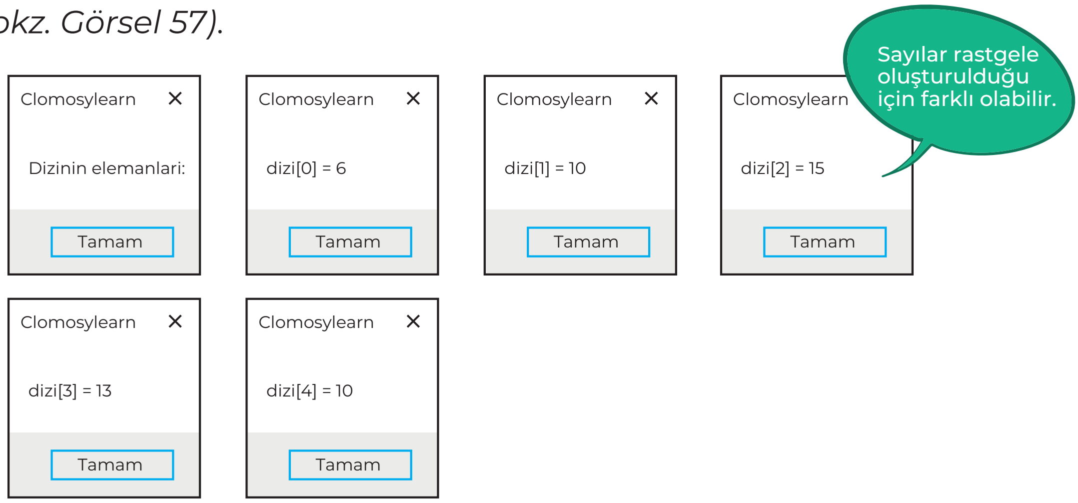
Ana Kod

```
var
 Dizi: array[0..4] of Integer;
 i,j: Integer;
 {
 for (i = 0 to 4)
 {
   Dizi[i] = Random() * 30;
 }

 // Diziyi ekrana yazdırma
 ShowMessage('Dizinin elemanlari:');
 for (i = 0 to Length(Dizi) - 1)
   ShowMessage('dizi[' + IntToStr(i) + '] = '+ IntToStr(Dizi[i]));
}
```

Kod 45: Dizi Örneği 2

Kod 45'te 'DiziOrnek' adında bir prosedür tanımlandı ve uygulama başladığı anda çağrıldı. İçinde 'Dizi' adında 5 elemanlı bir tam sayı dizisi ve iki tam sayı değişkeni (i ve j) tanımlandı. İlk for döngüsünde 'i' değişkenine 0'dan 4'e kadar tekrar yapılarak dizinin her bir elemanına 0 ile 30 arasında rastgele bir değer atandı. Ardından, ShowMessage fonksiyonu ile "Dizinin elemanları:" mesajı ekranda gösterildi. İkinci for döngüsünde 'i' değişkenine 0'dan dizinin son indisine kadar tekrar yapılarak dizinin her bir elemanının değeri ve indisi ekranda gösterildi. Diziye rastgele değerler atandı ve bu değerler ekranda sırasıyla gösterildi. *(bkz. Görsel 57)*.



Görsel 57: Örnek 2 Sonucu

Örnek 3

Bir dizi tanımlayalım ve bu dizinin elemanlarını küçükten büyüğe doğru sıralayalım. *Kod 46*'da 'Dizi' adında 4 elemanlı bir tam sayı dizisi tanımlandı ve sırasıyla 42, 34, 81, 10 değerleri atandı. İlk for döngüsü kullanılarak dizinin sıralanmadan önceki hali *Console.Text* fonksiyonu ile ekrana yazdırıldı. Ardından, seçim sıralama algoritması uygulandı. Bu algoritmada, 'i' değişkeni ile dizi üzerinde tekrar yapıldı ve her tekrarda 'Min' değişkeni ile en küçük elemanın indisi belirlendi. İç içe for döngüsünde, 'j' değişkeni ile sıralanmamış bölümdeki elemanlar karşılaştırılır ve daha küçük bir eleman bulunursa bu elemanın indisi 'Min' değişkenine atanır. Bulunan en küçük eleman, sıralanmamış bölümün ilk elemanı ile yer değiştirir. Bu işlem, dizinin tamamı sıralanana kadar devam eder. Son olarak, sıralanmış dizi Console.Text fonksiyonu ile ekrana yazdırılır.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum7/Ornek3)

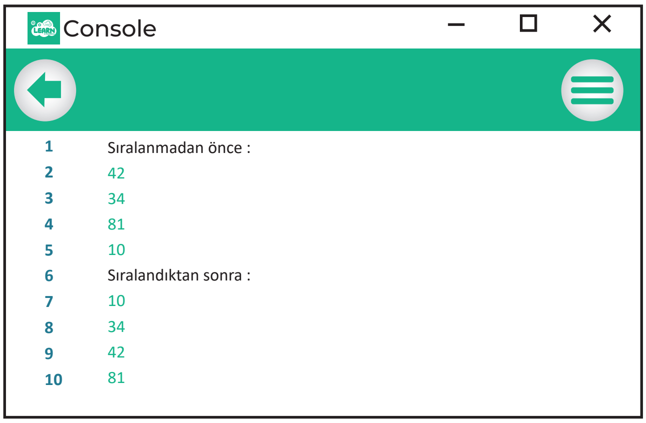
Ana Kod

Windows uygulamasında çalıştırarak deneyebilirsiniz.

```
var
  Dizi: array[3] of Integer;
  i, j, Min, Tut: Integer;
{
   //Diziyi tanımla
   Dizi[0] = 42;
   Dizi[1] = 34;
   Dizi[2] = 81;
   Dizi[3] = 10;
   Console.Text('Sıralanmadan önce: '); //Sıralanmadan önce diziyi yazdır
 for (i = 0 to Length(Dizi)-1)
    Console.Text(IntToStr(Dizi[i]));
    //Seçim sıralama algoritması
 for (i = 0 to Length(Dizi)-1)
 {
    Min = i;
   for (j = i + 1 to Length(Dizi)-1)
   {
    if (Dizi[j] < Dizi[Min])
    {
      Min = j;
      //En küçük elemanı, sıralanmamış bölümün ilk elemanıyla yer değiştir
      Tut = Dizi[i];
      Dizi[i] = Dizi[Min];
      Dizi[Min] = Tut;
    }
   }
 }
    //Sıralandıktan sonra diziyi yazdır
    Console.Text('Sıralandıktan sonra: ');
 for (i = 0 to Length(Dizi)-1)
    Console.Text(IntToStr(Dizi[i]));
}
```

Kod 46: Dizi Örneği 3

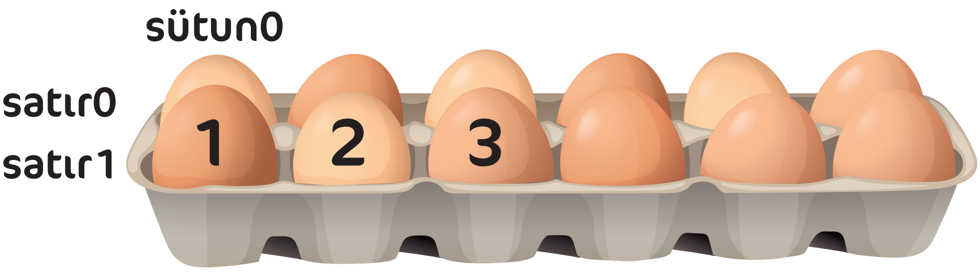
Görsel 58'de çıktısı yer almaktadır.



Görsel 58: Örnek 3 Sonucu

ÇOK BOYUTLU DİNAMİK DİZİ

Bu yapı, matris olarak oluşturulabilen dizilerden oluşur. Yumurta kabındaki her sıra, yumurta kabı dizisinin bir elemanı olarak düşünülebilir ve her sıranın içinde yumurtaları temsil eden elemanlar yer alır. Bu durumda, yumurta kabı çok boyutlu bir diziye benzetilebilir.



Clomosy'de çok boyutlu dinamik dizi tanımlamak için aşağıdaki söz dizimi kullanılır:

```
var
    CokBoyutluDizi;
```

Buna göre ismi 'CokBoyutluDizi' olarak belirlenen ve integer türünde veriler tutabilecek iki boyutlu dinamik dizi oluşturulur. Bu dizi, başlangıçta boştur ve boyutları çalışma zamanında belirlenebilir. Tanımlama yapıldıktan sonra ölçülerin belirtilmesi gerekir. *VarArrayCreate()* fonksiyonu, boyut oluşturmak için kullanılır. Bu fonksiyon iki parametre alır.

```
function VarArrayCreate(const Bounds: array of Integer; VarType: Integer): Variant;
```

Bounds: array of Integer: Bu parametre, oluşturulan çok boyutlu dizinin boyutlarını belirtir. Dinamik bir dizidir ve her boyut için minimum ve maksimum indis değerlerini içerir. Örneğin; [0, 2, 0, 2] ifadesi iki boyutlu bir dizi oluşturur. Birinci boyutun 0'dan 2'ye, ikinci boyutun da 0'dan 2'ye kadar indisleri vardır.

VarType: Integer: Bu parametre, dizideki ögelerin veri tipinin tanımlanmasını sağlar. Veri tipi bir tam sayı ile kodlanır. Örneğin: VarType değeri 3 kullanıldığında dizi içerisindeki ögelerin veri tipi integer olarak tutulur. Diğer veri tipleri *Tablo 7*'de görülebilir.

Type	VarType
varInteger	3
varDouble	5
varVariant	12
varString	256

Tablo 7: Çok Boyutlu Dizi Türleri

Örnek 1

Tablo 8'de dört satır, üç sütundan oluşan iki boyutlu bir dizi tanımlandı. *VarArrayCreate* fonksiyonunun isminden sonra köşeli parantez içine yazılan "0,3" ifadesi dizide 0'dan 3'e kadar numaralandırılan 4 satır olacağını ifade eder. Sonrasında gelen "0,2" ifadesi de, benzer şekilde 0'dan 2'ye kadar numaralandırılarak 3 sütun olacağını ifade eder. Son olarak eklenen 12 ise değişken türünün kodudur. *Tablo 7* kontrol edildiğinde dizinin değişken türünün *varVariant* olarak belirlendiği görülebilir.

	Sütun (Column)-0	Sütun (Column)-1	Sütun (Column)-2
Satır (Row) - 0	CokBoyutuDizi[0,0]	CokBoyutluDizi[0,1]	CokBoyutluDizi[0,2]
Satır (Row) - 1	CokBoyutluDizi[1,0]	CokBoyutluDizi[1,1]	CokBoyutluDizi[1,2]
Satır (Row) - 2	CokBoyutluDizi[2,0]	CokBoyutluDizi[2,1]	CokBoyutluDizi[2,2]
Satır (Row) - 3	CokBoyutluDizi[3,0]	CokBoyutluDizi[3,1]	CokBoyutluDizi[3,2]

Tablo 8: Örnek 1 Sonucu

CokBoyutluDizi[3,0]

Dizinin Adı — Satır İndisi — Sütun İndisi

BİLGİLENDİRME:
Çok boyutlu dizinin elemanlarına değer ataması yaparken veya elemanları çağırırken önce satır indisi sonra sütun indisi yazılır: dizi[satır,sütun]

Örnek 2

Oluşturulan diziye eleman ataması yaparak dizinin elemanlarına "CokBoyutluDizi[0,0] = 1" gibi ifadelerle tek tek değer atanır. Fakat *Kod 47*'de, dizinin tüm elemanlarına tek satırda değer atandı.

Ana Kod

```
var
   CokBoyutluDizi;
{
   CokBoyutluDizi = VarArrayCreate([0, 3, 0, 2], 12);// 4X3'lük dizi

   CokBoyutluDizi = [[1, 2, 3], [5, 6, 7], [9, 10, 11], [13, 14, 15]];
}
```

Kod 47: Çok Boyutlu Dizi Örneği 2

Kod 47'deki dizi *Tablo 9*'da açıklanmıştır. Dizinin ilk satırına 1, 2, 3 değerleri atandı. Yani 0.satırın, 0.sütununda 1 değeri saklandı. Farklı olarak dizinin [0,1] indisinde 2 değeri saklandı. Benzer şekilde sonraki satıra 5, 6, 7 değerleri atandı. Bu şekilde devam ederek 2 boyutlu bir dizi oluşturuldu.

	Sütun (Column)-0	Sütun (Column)-1	Sütun (Column)-2
Satır (Row) - 0	1	2	3
Satır (Row) - 1	5	6	7
Satır (Row) - 2	9	10	11
Satır (Row) - 3	13	14	15

Tablo 9: Örnek 2 Sonucu

CokBoyutluDizi[3,0]   CokBoyutluDizi[2,1]

Örnek 3

Örnekte 4x3 boyutunda çok boyutlu dizi oluşturuldu. Diziye değerler atandı. Oluşturulan dizinin elemanları ekranda gösterildi *(bkz. Kod 48).*

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum7/Ornek4)

Ana Kod

```
var
  Matris : Variant ;
  i, j,satirSayisi,sutunSayisi: Integer;
{
  Matris = VarArrayCreate([0,3,0,2],12);
  Matris = [[1, 2, 3], [5, 6, 7], [9, 10, 11], [13, 14, 15]];

  sutunSayisi = Length(Matris[0]); //4 satır
  satirSayisi = Length(Matris); //3 sütun
  ShowMessage(Matris[3,0]);
  for (i = 0 to satirSayisi-1)
  {
    for (j = 0 to sutunSayisi-1)
      ShowMessage(IntToStr(Matris[i, j]));
      ShowMessage(' ');
  }
}
```

Kod 48: Çok Boyutlu Dizi Örneği 3

Örnek 4

Örnekte, 3x3 boyutunda çok boyutlu dizi oluşturuldu. Oluşturulan dizinin her bir elemanına sırasıyla, (satır+1) * (sütun+1) değeri atandı. Oluşturulan dizi konsola yazdırıldı *(bkz. Kod 49).*

Oluşan dizi *Tablo 10*'da gösterilmektedir.

	0.Index	0.Index	0.Index
0.Index	1	2	3
1.Index	2	4	6
2.Index	3	6	9

(satır+1) * (sütun+1)

(1+1) * (2+1)

Tablo 10: Örnek 4 Dizi Yapısı

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum7/Ornek5)

Ana Kod

```
var
  Matris ;
  i, j: Integer;
  Tut : String;
{
  Tut = '';
  Matris = VarArrayCreate([0,2,0,2],12);
  for (i = 0 to 2)
    for (j = 0 to 2)
      Matris[i, j] = (i + 1) * (j + 1);
      ShowMessage('Matrisin Elemanlari:');

  for (i = 0 to 2)
  {
    for (j = 0 to 2)
      Tut = Tut +' '+IntToStr(Matris[i, j])
      ShowMessage (Tut);
      ShowMessage(' ');
      Tut = '';
  }
}
```

Kod 49: Çok Boyutlu Dizi Örneği 4

ÖZELLEŞTİRİLMİŞ DİZİLER (TCLARRAY)

TclArray, Clomosy platformunun sunmuş olduğu özelleştirilmiş dizi tanımlamasıdır. TclArray sınıfından bir diziyi oluştururken dizinin eleman sayısını belirlemek gerekmez. Dizinin boyutu, eleman ataması yaptıkça dinamik olarak belirlenir.

TclArray sınıfına ait farklı veri türlerini kullanabilmek için türetilmiş (kalıtım) sınıflar bulunur.

- TclArrayInteger
- TclArrayString
- TclArrayDouble

Değişken tanımlaması şu şekildedir:

```
var
  Dizi:TclArrayInteger;
```

Bu dizi, standart dizi kullanımı dışında, dinamik dizi oluşturulması için kullanılır. Tanımlanan değişkenin dizi olarak kullanılması için nesne oluşturulmalıdır.

```
var
  Dizi:TclArrayInteger;
{
    Dizi= TclArrayInteger.Create;
}
```

Diziye değer ekleme işlemini gerçekleştirebilmek için *Add* fonksiyonu kullanılır. *Count* fonksiyonu, dizi içerisindeki toplam öge sayısını öğrenebilmek için kullanılır *(bkz. Kod 50).*

Ana Kod

```
var
 Dizi : TClArrayInteger;
{
   Dizi = TClArrayInteger.Create;
   Dizi.Add(20);
   Dizi.Add(30);
   Dizi.Add(17);
   Dizi.Add(23);
   ShowMessage(Dizi.Count);
}
```

Kod 50: TclArray Kullanımı

Dizinin öğelerine erişmek için *GetItem(index)* fonksiyonu kullanılır. *SetItem(index,newVaue)* ise belirli bir dizideki belirli bir indise sahip ögenin değerini değiştirmek için kullanılır.

```
Dizi.GetItem(index)
Dizi.SetItem(index,newValue)
```

- *index:* Değiştirmek istediğiniz ögenin dizideki indisi.
- *newValue:* Değiştirmek istediğiniz ögenin yeni değeri.

Örnek 1

Kod 51'de 'Dizi' isimli özelleştirilmiş dizi tanımlandı. Bu diziye 4 eleman eklenip ekrana yazdırıldı. Daha sonra dizinin 2.indisindeki eleman değiştirilip dizi ekrana tekrar yazdırıldı.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum7/Ornek6)

Ana Kod

```
var
 Dizi : TClArrayInteger;
 i: Integer;
{
   Dizi = TClArrayInteger.Create;
   Dizi.Add(20);
   Dizi.Add(30);
   Dizi.Add(17);
   Dizi.Add(23);
   ShowMessage(Dizi.Count);
 for (i = 0 to Dizi.Count-1)
 {
   ShowMessage('Değer: '+IntToStr(Dizi.GetItem(i)));
 }

   Dizi.SetItem(2,88);

   ShowMessage('Dizinin yeni hali');
 for (i = 0 to Dizi.Count-1)
 {
   ShowMessage ('Değer: '+IntToStr(Dizi.GetItem(i)));
 }
}
```

Kod 51: TclArray Örneği 1

Dizinin ilk hâli

Dizi.GetItem(0)	Dizi.GetItem(1)	Dizi.GetItem(2)	Dizi.GetItem(3)
20	30	17	23

Dizi.SetItem(2,88) işleminden sonraki hâli

Dizi.GetItem(0)	Dizi.GetItem(1)	Dizi.GetItem(2)	Dizi.GetItem(3)
20	30	88	23

Destroy yöntemi, dinamik olarak oluşturulmuş dizi nesnesinin bellekten temizlenmesi için kullanılır. Bu şekilde, bellek sızıntıları ve gereksiz bellek kullanımı önlenmiş olur.

```
Dizi.Destroy;
```

Bir dizideki tüm ögeleri kaldırmak veya silmek için *RemoveAll* yöntemi kullanılır. *RemoveAt()* metodu, bir diziden belirtilen bir indisteki ögeyi silmek için kullanılır. Bu metod silme işleminden sonra dizinin boyutunu otomatik olarak yeniden düzenler.

```
Dizi.RemoveAll;
Dizi.RemoveAt(index)
```

- *index:* Kaldırılacak ögenin indisidir. Dizideki hangi konumda bulunan ögenin kaldırılacağını belirtir.

Örnek 2

Kod 52'de 'Dizi' isimli özelleştirilmiş dizi tanımlandı. Bu diziye 4 eleman eklenip ekrana yazdırıldı. Daha sonra dizinin 0.indisindeki eleman silinip dizi ekrana tekrar yazdırıldı.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum7/Ornek7)

Ana Kod

```
var
  Dizi : TClArrayInteger;
   i: Integer;
{
  Dizi = TClArrayInteger.Create;
  Dizi.Add(20);
  Dizi.Add(30);
  Dizi.Add(17);
  Dizi.Add(23);
  for (i = 0 to Dizi.Count-1)
  {
     ShowMessage('Değer: '+IntToStr(Dizi.GetItem(i)));
  }
     Dizi.RemoveAt(0);

     ShowMessage('Silme işleminden sonraki yeni hali');
  for (i = 0 to Dizi.Count-1)
  {
     ShowMessage('Değer: '+IntToStr(Dizi.GetItem(i)));
  }
 }
```

Kod 52: TclArray Örneği 2

Dizinin ilk hâli

Dizi.GetItem(0)	Dizi.GetItem(1)	Dizi.GetItem(2)	Dizi.GetItem(3)
20	30	17	23

Dizi.RemoveAt(0) ifadesinden sonraki hâli:

Dizi.GetItem(0)	Dizi.GetItem(1)	Dizi.GetItem(2)
30	17	23

Örnek 3

Bir dizi tanımlandı ve elemanları büyükten küçüğe sıralandı *(bkz. Kod 53).*

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum7/Ornek8)

Ana Kod

```
var
 Dizi : TClArrayInteger;
 i,j,Max,Tut, Tut_ind: Integer;

{
    Dizi = TClArrayInteger.Create;
    Dizi.Add(20);
    Dizi.Add(30);
    Dizi.Add(17);
    Dizi.Add(23);

    //Sıralanmadan önce diziyi yazdır
    ShowMessage('Sıralanmadan önce: ');
 for (i = 0 to Dizi.count-1)
     ShowMessage(IntToStr(Dizi.GetItem(i)));

 for (i = 0 to Dizi.count-2)
 {
     Max=Dizi.GetItem(i);
  for(j=i+1 to Dizi.count-1)
  {
   if(Dizi.GetItem(j)>Max)
   {
     Max=Dizi.GetItem(j);
     Tut_ind=j;
   }
  }
     Tut=Dizi.GetItem(i);
     Dizi.SetItem(i,Max);
     Dizi.SetItem(Tut_ind,Tut);
     Max=0;
 }

     //Sıralandıktan sonra diziyi yazdır
     ShowMessage('Sıralandıktan sonra: ');
 for (i = 0 to Dizi.count-1)
     ShowMessage(IntToStr(Dizi.GetItem(i)));
}
```

Kod 53: TclArray Örneği 3

Alıştırmalar

QR OKUT

1. Soru ve cevapların beraber tutulabileceği bir dizi yapısı oluşturunuz. Bu yapıda birden fazla soru ve cevabı tutarak her birini ekranda gösterecek bir program yazınız.

2. Bir dizi tanımlayınız ve indislerine 1'den 10'a kadar olan sayıların karelerini atayınız. Sonrasında bu diziyi bir for döngüsüyle ekrana yazdırınız.

3. Bir dizi tanımlanmış ve bu dizinin elemanları bir for döngüsü kullanılarak ekrana yazdırılmak istenmiştir. Aşağıdaki kod parçasında bazı eksiklikler ve yazım hataları bulunmaktadır:

```
var
  kontrol: array[3] of Integer;
  i: integer;
  {
  kontrol = [1, 2, 3];
  for (i = 1 to length(kontrol) - 1) {
     ShowMessage(kntrol[i]);
  }
  }
```

Bu kod parçasındaki eksiklikleri ve hataları tespit ederek düzeltiniz. Kodun doğru ve çalışır bir hâlini yazınız.

4. 6 elemanlı Integer türünde dizi oluşturunuz. Bu dizinin elemanlarının toplamını hesaplayıp ekrana yazdırınız.

5. Mevsimleri içeren dizi tanımlayıp dizinin elemanlarını ekrana yazdırınız.

6. Tek boyutlu bir sayı dizisi oluşturunuz. Dizi, rastgele değerler içeren 5 elemandan oluşacaktır. Bu dizideki en küçük elemanı bulunuz ve bu elemanın yerine 1 değerini atayınız.

https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum7

Ek Kaynaklar

- Diziler (Tek Boyutlu ve Çok Boyutlu)
  https://www.docs.clomosy.com/Arrays
- Özelleştirilmiş Diziler (TclArray)
  https://www.docs.clomosy.com/TclArray

BÖLÜM 8: PROSEDÜR(VOID) FONKSİYONLAR VE HAZIR KÜTÜPHANE FONKSİYONLARI

PROSEDÜR (VOID)

Kodlar geliştikçe genellikle daha uzun ve karmaşık bir yapıya dönüşür. Bu durumda, programları daha etkin bir şekilde yönetebilmek için onları daha küçük, yönetilebilir parçalara bölmek faydalı olur; böylece kodun okunabilirliği artar ve bakımı önemli ölçüde kolaylaşır. İşte bu noktada prosedürler devreye girer. Prosedürler, kodları modüler bir şekilde organize etmeyi sağlayan yeniden kullanılabilir kod bloklarıdır.

Bu bloklar, kullanılacakları yerden önce tanımlanır; böylece programın ilerleyen kısımlarında gerektiğinde çağrılabilir ve kullanılabilir.

Prosedür bir kahve makinesine benzetilebilir. Örneğin; *espresso* yapmak için bir düğmeye basıldığında kahve makinesi kahveyi öğütme, sıcak suyu belirli bir basınçla kahve üzerinden geçirme ve son olarak kahveyi fincana dökme gibi adımları otomatik olarak gerçekleştirir. Bu, yazılımda bir prosedürün çağrılması gibidir. Prosedür, tanımlandığı işlemleri adım adım uygular ve istenen sonucu üretir. Programın belirli bir noktasında çağrıldığında tanımlanan görevleri yerine getirir ve ardından programın geri kalan kısmı çalışmaya devam eder.

Prosedür, *void* anahtar kelimesi ile tanımlanır.

Void şu şekilde tanımlanır:

```
void <void_ismi>;
<yerel bildirimler>
{
  //istenilen ifadeler yazılır
}
```

Prosedür (void) parametreli ve parametresiz olarak tanımlanabilir. İlk olarak parametresiz prosedürü inceleyelim

PARAMETRESİZ PROSEDÜR

Parametresiz prosedürler, belirli bir işlem veya işlemler dizisini gerçekleştirmek için kullanılan ve isimleri ile çağırılan kısa yollar gibidir.

Kod 54'te 'MesajYazdir' isimli bir prosedür tanımlandı ve ana kod bloğu içinde çağırılarak ekrana mesaj yazdırması sağlandı. Main blokları içerisinde, tanımlı olan prosedür çağrıldı. Bu çağrılan prosedür içerisinde hangi görev yapıldıysa o görevler işlendi.

Ana Kod

```
void MesajYazdir;
{
   ShowMessage('Bu bir parametresiz void yapısıdır.');
}

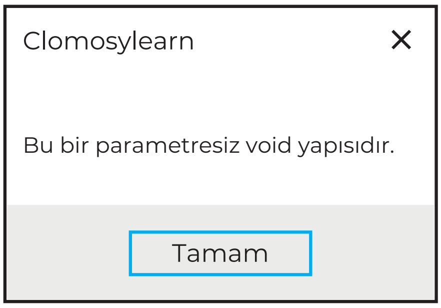
{
   MesajYazdir;
}
```

Prosedürün tanımlanması (satır 1–4)

Prosedürün ana kod bloğunda çağırılması (satır 6–8)

Kod 54: Parametresiz Prosedür Kullanımı

Prosedürün ekran çıktısı *Görsel 59*'daki gibi olacaktır.



Görsel 59: Prosedür Sonucu

Örnek 1

Bir sayının tek mi çift mi olduğunu ayırt eden bir prosedür tanımlandı ve ana kod bloğu içinde çağırılarak çalıştırıldı *(bkz. Kod 55).*

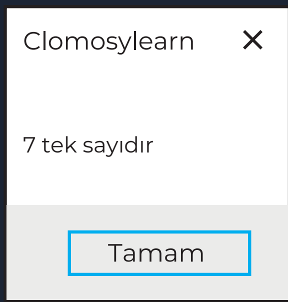
QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum8/Ornek1)

Ana Kod

```
// Prosedürün tanımlanması
void TekmiCiftmi;
var
 Sayi: Integer;
{
 Sayi=7;
 // Tek mi çift mi kontrolü
 if(Sayi mod 2 == 0)
 {
  ShowMessage(IntToStr(Sayi)+' çift sayıdır');
 }
 else
 {
  ShowMessage(IntToStr(Sayi)+' tek sayıdır');
 }
}

{
 // TekmiCiftmi prosedürünü çağır
 TekmiCiftmi();
}
```



Kod 55: Parametresiz Prosedür Örneği 1

Kod 55'te, 'TekmiCiftmi' adında bir prosedür tanımlandı ve uygulama açıldığı anda bu prosedüre gitti. Prosedür içinde bir tam sayı olan 'Sayi' değişkeni değerine 7 atandı. Prosedür, 'Sayi' değişkeninin tek mi yoksa çift mi olduğunu kontrol etti. If bloğunda, "Sayi mod 2 == 0" kontrolü yapıldı; bu, 'Sayi' değişkeni 2'ye tam bölünüyorsa (mod operatörü tam sayı bölmesi yapar), sayının çift olduğunu belirtti ve ekrana ShowMessage ile "Sayi çift sayıdır" mesajı gösterilir. Aksi takdirde, else bloğu çalışır ve ekrana ShowMessage ile "Sayi tek sayıdır" mesajı gösterir.

Örnek 2

Dikdörtgenin kısa kenar uzunluğu ile uzun kenar uzunluğunu kullanarak dikdörtgenin alanını hesaplayan bir prosedür tanımlandı. Prosedür ‘KisaKenar’ ve ‘UzunKenar’ değişkenlerini çarparak elde edilen değeri, ‘Alan’ değişkenine atama yaptı daha sonra da bu değeri ekrana yazdırdı *(bkz. Kod 56)*.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum8/Ornek2)

Ana Kod

```
// Prosedürün tanımlanması
void AlanHesapla;
var
  KisaKenar, UzunKenar, Alan : Integer;
{
  KisaKenar = 5;
  UzunKenar = 7;

  // Alan hesaplama
  Alan = KisaKenar * UzunKenar;

  // Sonucu ekrana yazdir
  ShowMessage(' dikdörtgenin alanı: '+ IntToStr(Alan));
}

{
  // Alan hesaplama prosedürünü çağır
  AlanHesapla();
}
```

Clomosylearn

dikdörtgenin alanı: 35

Tamam

Kod 56: Parametresiz Prosedür Örneği 2

Örnek 3

Kod 57'de, bir sayının faktöriyelini hesaplayan 'FaktoriyelHesapla' adında bir prosedür tanımlandı. Prosedürde 'i', 'Sayi' ve 'Faktoriyel' adında üç tam sayı değişkeni bulunmaktadır. Başlangıçta 'Faktoriyel' değişken değerine 1 ve 'Sayi' değişken değerine 5 atandı. Bir for döngüsü kullanılarak, 1'den başlayıp 'Sayi' değişkeninin değeri olan 5'e kadar tüm tam sayılar için 'Faktoriyel' değişkeni 'i' ile çarpıldı. Bu işlem sayesinde sayının faktöriyeli hesaplandı. Hesaplama tamamlandığında, ShowMessage fonksiyonu ile 'Sayi' değişkeninin faktöriyeli ekranda gösterildi. Ana programda, 'FaktoriyelHesapla' prosedürü çağrılarak faktöriyel hesaplama işlemi gerçekleştirildi.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum8/Ornek3)

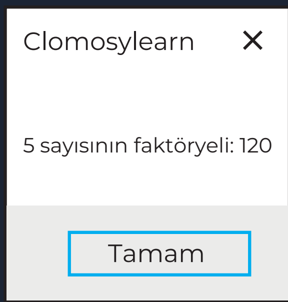
Ana Kod

```
/// Prosedürün tanımlanması
void FaktoriyelHesapla;
var
  i, Sayi, Faktoriyel: Integer;
{
  Faktoriyel = 1;
  Sayi = 5;

  // Faktöriyel hesaplama
  for (i = 1 to Sayi)
  Faktoriyel = Faktoriyel * i;

  // Sonucu ekrana yazdır
  ShowMessage(IntToStr(Sayi) + ' sayısının faktöriyeli: '+ IntToStr(Faktoriyel));
}

{
  // Faktöriyel hesaplama prosedürünü çağır
  FaktoriyelHesapla();
}
```



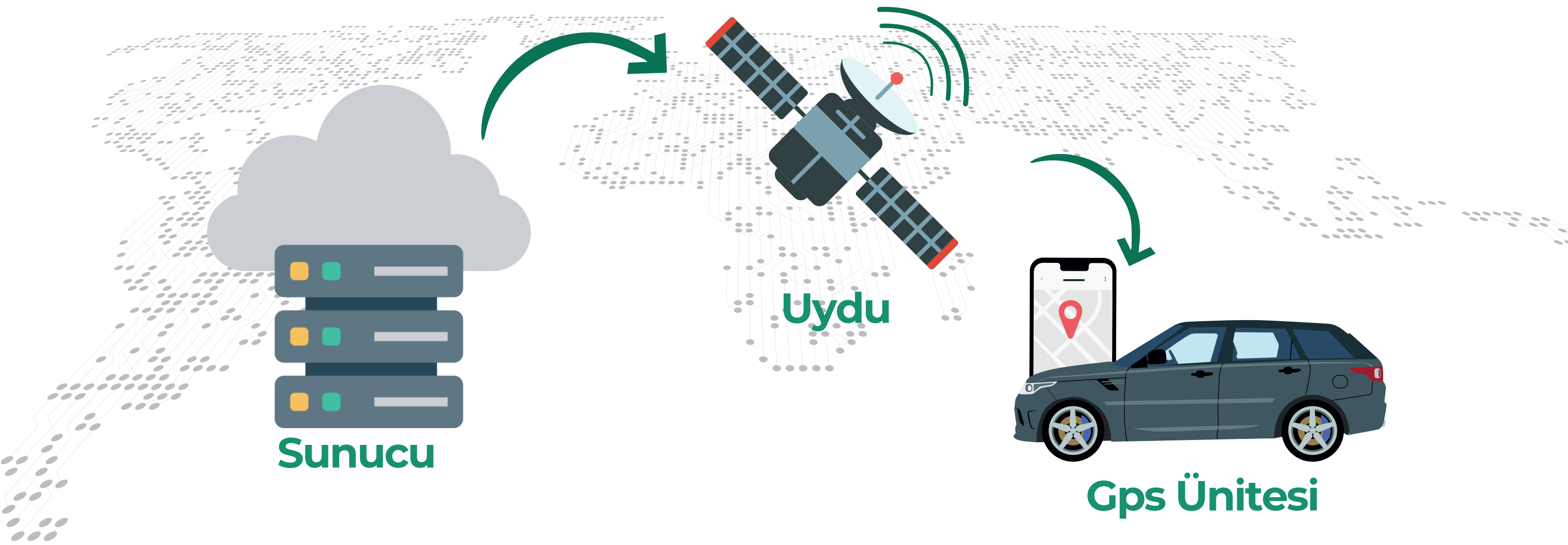
Kod 57: Parametresiz Prosedür Örneği 3

PARAMETRELİ PROSEDÜR

Prosedürler parametreli olarak da tanımlanabilir. Bu durumda çalışması için bazı başlangıç değerleri verilmelidir. Bir önceki kahve makinesi örneği ele alındığında kahvenin hazırlanması için gerekli kahve çekirdeği, su, şeker gibi malzemeler parametreler olarak düşünülebilir.

Örneğin, TOGG'un elektrikli aracının yazılımı, internet üzerinden güncellenebilir. Araç, yazılım güncellemesi alabilmek için önce sunucuyla iletişime geçer. Aracın mevcut yazılım versiyonu, sunucudan gelen en son versiyonla karşılaştırılır. Eğer aracın versiyonu eskiyse yeni güncelleme için araç otomatik olarak indirme işlemini başlatır.

Örnekte, aracın mevcut yazılım versiyonu ve sunucudan alınan yeni versiyon bilgisi parametre olarak prosedüre iletilir. Prosedür içerisinde ise aracın yazılım versiyonu kontrol edilir ve gerekirse güncelleme başlatılır.



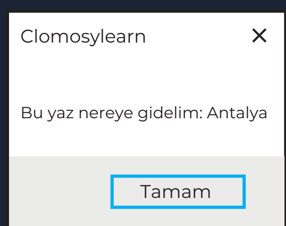
Örnek 1

Kod 58'de 'Tatile' isimli bir prosedür tanımlandı. Prosedür 'Belde' isimli string türünde bir parametre alır ve bunu ShowMessage içinde kullanır.

Ana Kod

```
void Tatile(Belde: String);
{
 ShowMessage('Bu yaz nereye gidelim: '+Belde);
}

{
 Tatile('Antalya');
}
```



Kod 58: Parametreli Prosedür Örneği 1

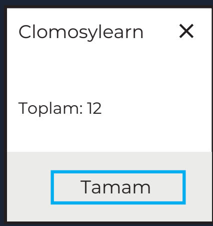
Örnek 2

Kod 59'da 'Toplam' isimli bir prosedür tanımlandı. Prosedür 'Sayi1' ve 'Sayi2' isimli iki integer değeri parametre olarak alır ve toplamlarını ekrana yazdırır.

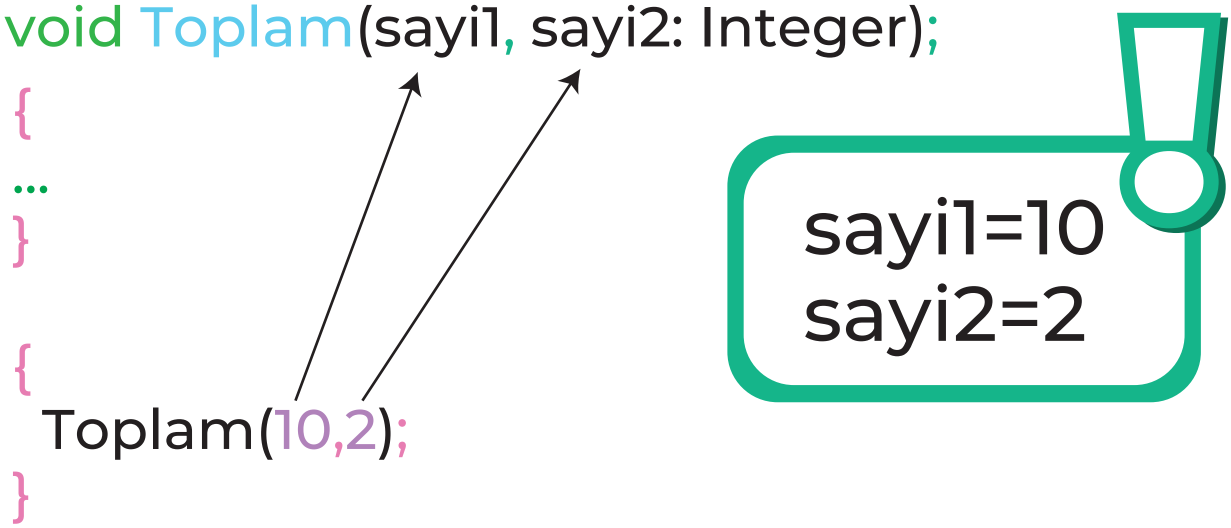
Ana Kod

```
void Toplam(Sayi1, Sayi2: Integer);
var
 Cevap: Integer;
{
 Cevap = Sayi1 + Sayi2;
 ShowMessage('Toplam: ' + IntToStr(Cevap));
}

{
 Toplam(10,2);
}
```



Kod 59: Parametreli Prosedür Örneği 2



Görsel 60: Toplam Prosedürünün Yapısı

Örnek 3

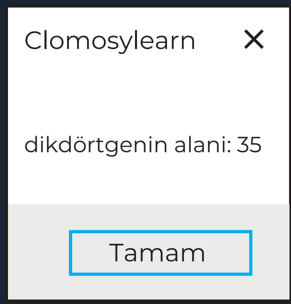
Dikdörtgenin kısa kenar uzunluğu ile uzun kenar uzunluğu parametre olarak alındı, dikdörtgenin alanını hesaplayan bir prosedür yazıldı *(bkz. Kod 60).*

Ana Kod

```
// Prosedürün tanımlanması
void AlanHesapla(KisaKenar, UzunKenar: Integer);
var
 Alan: Integer;
{
 // Alan hesaplama
 Alan = KisaKenar * UzunKenar;

 // Sonucu ekrana yazdır
 ShowMessage(' dikdörtgenin alani: '+ IntToStr(Alan));
}

{
 // Alan hesaplama prosedürünü çağır
 AlanHesapla(5,7);
}
```



Kod 60: Parametreli Prosedür Örneği 3

Örnek 4

Bu örnekte 'FaktoriyelHesapla' prosedürü, parametreli olarak tanımlandı. Prosedür 'Sayi' değişkenini parametre olarak aldı *(bkz. Kod 61).*

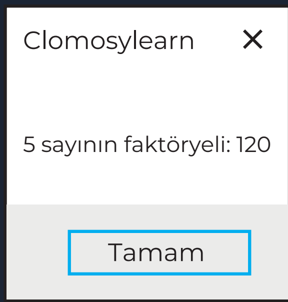
QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum8/Ornek4)

Ana Kod

```
// Parametreli bir prosedür tanımı
void FaktoriyelHesapla(Sayi: Integer);
var
  i, Faktoriyel: Integer;
{
  Faktoriyel = 1;
  // Faktöriyel hesaplama
  for (i = 1 to Sayi)
  Faktoriyel = Faktoriyel * i;

  // Sonucu ekrana yazdır
  ShowMessage(IntToStr(Sayi) + ' sayısının faktöriyeli: '+ IntToStr(Faktoriyel));
}
{
  // Faktöriyel hesaplama prosedürünü çağır
  FaktoriyelHesapla(5);
}
```



Kod 61: Parametreli Prosedür Örneği 4

FONKSİYON (FUNCTION)

Fonksiyonlar, prosedürlerle benzer amaçlar için kullanılır ve çalışma prensipleri açısından onlara oldukça benzerdir.

Ancak, fonksiyonların temel farkı, işlem sonucunda bir değer döndürmeleridir.

Yine kahve makinesinden örnek verildiğinde makinenin hazır olarak bize sunduğu kahve, fonksiyonun döndürdüğü değere benzetilebilir.

Ya da bir otomatın, ödenilen paraya karşılık kişiye verdiği ürün, döndürdüğü değer olarak düşünülebilir. Fonksiyonlar tanımlanırken *function* anahtar kelimesi kullanılır.

Function şu şekilde tanımlanır:

```
Function <fonksiyon_ismi> (<parametre listesi>): <dönüş_değeri_tipi>;
<yerel bildirimler>
{
   //istenilen ifadeler yazılır
}
```

BİLGİLENDİRME:
Result, bir fonksiyonun geri döndüreceği değeri belirlemek için kullanılan özel bir anahtar kelimedir. Bir fonksiyon içinde Result değişkenine atanan değer, fonksiyonun çağrıldığı yere döndürülür. Result, fonksiyonun geri dönüş tipine uygun bir değişken gibi davranır ve fonksiyonun dönüş tipine uygun bir değeri saklar.

Örnek 1

Kod 62'de 'Karesi' isimli bir fonksiyon tanımlandı. Fonksiyon 'Sayi' isimli tam sayı türünde bir değişkeni parametre olarak alır ve Result anahtar kelimesini kullanarak bu sayının karesini yine bir tam sayı olarak döndürür. Fonksiyonun isimlendirildiği satırın sonundaki Integer ifadesi, fonksiyonun geri dönüş değerinin integer olduğunu ifade eder.

Ana Kod

```
//Fonksiyonun tanımlanması
function Karesi(Sayi: Integer): Integer;
{
 Result = Sayi * Sayi;
}

//Fonksiyonun ana kod bloğunda çağırılması
{
 ShowMessage(Karesi(6));
}
```

Clomosylearn ✕
36
Tamam

Kod 62: Fonksiyon Örneği 1

Örnek 2

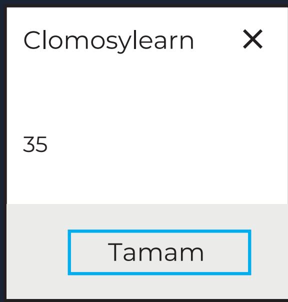
Dikdörtgenin kısa kenar uzunluğu ile uzun kenar uzunluğu parametre olarak alındı, dikdörtgenin alanını döndüren bir fonksiyon yazıldı *(bkz. Kod 63).*

Ana Kod

```
// Fonksiyonun tanımlanması
function AlanHesapla(KisaKenar, UzunKenar: Integer):Integer;
var
   Alan: Integer;
{
  // Alan hesaplama
  Alan = KisaKenar * UzunKenar;

  // Sonucu döndür
  Result = Alan;
}

{
  // Alan hesaplama prosedürünü çağır
  ShowMessage(AlanHesapla(5,7));
}
```



Kod 63: Fonksiyon Örneği 2

Örnek 3

Bu örnekte önceki kısımlarda prosedür olarak tanımlanan ‘FaktoriyelHesapla’ bu sefer fonksiyon olarak kullanıldı. Fonksiyon ‘Sayi’ değişkenini parametre olarak aldı ve Result anahtar kelimesini kullanarak hesapladığı faktöriyel değerini döndürdü.

Kod 64’te diğer örnekten farklı olarak fonksiyon üzerinden gelen değer, bir değişkene atanarak ekranda gösterildi.

QR Okut

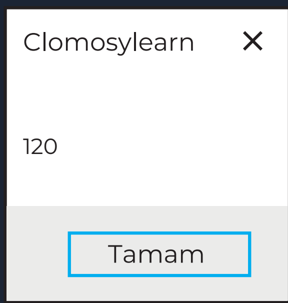
(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum8/Ornek5)

Ana Kod

```
var
  GelenDeger : Integer;
// Fonksiyon tanımı
Function FaktoriyelHesapla(Sayi: Integer): Integer;
var
  i, Faktoriyel: Integer;
{
  Faktoriyel = 1;
  // Faktöriyel hesaplama
  for (i = 1 to Sayi)
    Faktoriyel = Faktoriyel * i;

  // Sonucu döndür
  Result = Faktoriyel;
}

{
  // Faktöriyel hesaplama prosedürünü çağır
  GelenDeger = FaktoriyelHesapla(5);
  ShowMessage(GelenDeger);
}
```



Kod 64: Fonksiyon Örneği 3

HAZIR KÜTÜPHANE FONKSİYONLARI

Hazır kütüphane fonksiyonları, bir programlama dilinde belirli görevleri hızlı ve kolay bir şekilde gerçekleştirmek için önceden tanımlanmış fonksiyonlardır. Bu fonksiyonlar, yazılımcıların tekrar tekrar aynı kodu yazmak zorunda kalmadan, yaygın işlemleri yapmalarına olanak tanır. Hazır kütüphane fonksiyonları, programlama sürecini hızlandırır, kodun kalitesini artırır ve yazılımcıların daha verimli çalışmasına yardımcı olur.

Tasarlanan fonksiyonları kullanabilmek için ana kod bloğundan önce tanımlayıp nasıl çalışacaklarını planlamak gerekir. Ancak hazır kütüphane fonksiyonları, programlama dilini geliştiren yazılımcılar tarafından önceden tanımlanmıştır. Bu fonksiyonları kullanmak için kod bloğu içinde çağırmak yeterlidir.

Hazır kütüphane fonksiyonları, nereye gidileceği belirtildiğinde, yolculuk aşamalarını otomatik olarak planlayan navigasyon gibidir. Belirli bir görevi yerine getirmek üzere tasarlandıkları için müdahale etmeden istenen işlemi gerçekleştirir ve sonucunu bildirir. Böylece, fonksiyonların çalışma detaylarıyla uğraşmak zorunda kalmadan, ihtiyaç duyulan sonuçlar hızlı ve kolay bir şekilde elde edilir.



KARAKTER İŞLEME FONKSİYONLARI

Karakter işleme fonksiyonları, metin verilerini işlemek ve manipüle etmek için kullanılan güçlü araçlardır. Bu fonksiyonlar, programlama dilinde metinle ilgili çeşitli işlemlerin kolay ve hızlı bir şekilde gerçekleştirilmesini sağlar.

Mesela bir silgi, defterden yazılanları silebilir. *Delete* fonksiyonu da string veriler üzerinde silme işlemi yapabilir.

Tablo 11'de karakter işleme fonksiyonları için örnekler mevcuttur.

Fonksiyon	Özellik	Kullanımı
AnsiLowerCase	Tüm harflerin küçük harfe dönüştürüldüğü bir String kopyası oluşturur.	YeniMetin = AnsiLowerCase('CLOMOSY''i öğreniyorum.'); // clomosy'i öğreniyorum.
AnsiUpperCase()	Tüm harflerin büyük harfe dönüştürüldüğü bir String kopyası oluşturur.	YeniMetin = AnsiUpperCase('CLOMOSY''i öğreniyorum.'); // CLOMOSY'İ ÖĞRENİYORUM.
Copy()	String bir içeriğin kopyalanmasını sağlar. 3 parametre alır: 1.parametre, kaynak metni ifade eder. 2.parametre, kaçıncı karakterden başlayacağını belirtir. 3.parametre, kaç karakter alacağını belirtir.	HedefMetin = Copy('12345678', 3, 4); // 3456
Trim() TrimLeft() TrimRight()	Bir dizenin başındaki ve sonundaki boş ve kontrol karakterlerini (satır besleme gibi) kaldırır.	YeniMetin = Trim(' Clomosy '); // Clomosy
Delete()	String bir ifadenin istenilen kısmını silmektedir. 3 parametre alır: 1.parametre, kaynak metni ifade eder. 2.parametre, kaçıncı karakterden başlayacağını belirtir. 3.parametre, kaç karakter alacağını belirtir.	Delete('12345678', 3, 4); // 1278
Insert	Bir stringin belirli bir konumuna başka bir string değeri ekler. 3 parametre alır: 1.parametre, eklemek istenilen stringi ifade eder. 2.parametre, hedef string metnini ifade eder. 3.parametre, hedef stringin hangi konumuna ekleneceğini belli eder.	Insert(' dünya!',HedefMetin, 9); // HedefMetin => ilk hali = 'Merhaba,' // HedefMetin => son hali = 'Merhaba, dünya!'
AnsiCompareStr()	İki ANSI string ifadesini karşılaştıran bir işlevdir. Karşılaştırma büyük/küçük harfe duyarlıdır.	Sonuc = AnsiCompareStr('Clomosy', 'Clomosy'); // sonuç 0 olur.
AnsiCompareText()	İki ANSI string ifadesini karşılaştıran bir işlevdir. Karşılaştırma büyük/küçük harf farkını dikkate almaz.	Sonuc = AnsiCompareText('Clomosy', 'CLOMOSY'); // sonuç 0 olur

Tablo 11: Karakter İşleme Fonksiyonları

Örnek 1

Karakter işleme fonksiyonları basit bir şekilde kullanılarak bir metin işlendi ve her işlem açıklamalı bir şekilde mesaj penceresinde gösterildi *(bkz. Kod 65).*

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum8/Ornek6)

Ana Kod

```
var
  metin: String;
{
  metin = '  Merhaba Clomosy!  ';

  // 1. Küçük harfe çevir
  metin = AnsiLowerCase(metin);
  ShowMessage('Küçük harfe çevrildi: ' + metin); // Çıktı: '  merhaba clomosy!  '

  // 2. Büyük harfe çevir
  metin = AnsiUpperCase(metin);
  ShowMessage('Büyük harfe çevrildi: ' + metin); // Çıktı: '  MERHABA CLOMOSY!  '

  // 3. Baştaki ve sondaki boşlukları kaldır
  metin = Trim(metin);
  ShowMessage('Boşluklar kaldırıldı: ' + metin); // Çıktı: 'MERHABA CLOMOSY!'

  // 4. Belirli bir kısmı kopyala
  metin = Copy(metin, 1, 7); // İlk 7 karakteri al
  ShowMessage('İlk 7 karakter alındı: ' + metin); // Çıktı: 'MERHABA'

  // 5. Karakter sil
  Delete(metin, 4, 2); // 4. karakterden başlayarak 2 karakter sil
  ShowMessage('4. karakterden itibaren 2 karakter silindi: ' + metin); // Çıktı: 'MERBA'

  // 6. Metin ekle
  Insert('HA', metin, 4); // 4. pozisyona 'HA' ekle
  ShowMessage('4. pozisyona "HA" eklendi: ' + metin); // Çıktı: 'MERHABA'

  // 7. Metin karşılaştır (büyük-küçük harf duyarlı)
  if (AnsiCompareStr(metin, 'MERHABA') == 0)
    ShowMessage('Metin "MERHABA" ile birebir eşleşiyor (duyarlı).');

  // 8. Metin karşılaştır (büyük-küçük harf duyarsız)
  if (AnsiCompareText(metin, 'merhaba') == 0)
    ShowMessage('Metin "merhaba" ile birebir eşleşiyor (duyarsız).');
}
```

Kod 65: Karakter İşleme Fonksiyonları Örnek 1

Örnek 2

Karakter işleme fonksiyonlarından bir metnin uzunluğunu alan *Length* fonksiyonunu kullanan bir uygulama yapalım.

Bu örnekte 'TersCevir' adında bir fonksiyon tanımlandı ve bir string parametre (GelenMetin) aldı. Fonksiyon, 'GelenMetin' parametresinin tersini alarak geri döndürdü. Fonksiyon içinde bir for döngüsü kullanılarak 'GelenMetin' stringinin son karakterinden başlayarak ilk karakterine kadar olan tüm karakterler Result değişkenine eklendi. Result, fonksiyonun dönüş değeri olarak kullanılır ve ters çevrilmiş metni içerir. Ana kod bloğunda, metin adında bir string değişken tanımlandı ve "kitap" değeri atandı. ShowMessage fonksiyonu ile 'TersCevir (metin)' çağrıldı ve "kitap" kelimesinin tersi (patik) ekranda gösterildi *(bkz. Kod 66).*

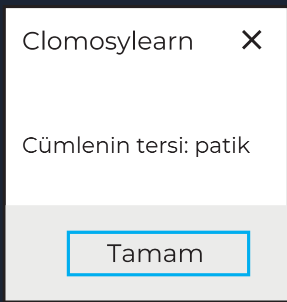
QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum8/Ornek7)

Ana Kod

```
function TersCevir(GelenMetin: string): string;
var
  i: Integer;
{
  Result = '';
  for (i = Length(GelenMetin) downto 1)
    Result = Result + GelenMetin[i];
}

var
  metin: string;
{
  metin = 'kitap';
  ShowMessage('Cümlenin tersi: '+ TersCevir(metin));
}
```



Kod 66: Karakter İşleme Fonksiyonu – Length

MATEMATİK FONKSİYONLARI

Matematik fonksiyonları, sayısal verileri işlemek ve matematiksel hesaplamalar yapmak için kullanılan temel araçlardır. Bu fonksiyonlar, programlama dilinde çeşitli matematiksel işlemlerin kolay ve hızlı bir şekilde gerçekleştirilmesini sağlar. Güçlü bir hesap makinesi gibidirler.

Tablo 12'de matematik fonksiyonları için örnekler yer alıyor.

Fonksiyon	Özellik	Kullanımı
Abs()	Bir tam sayının mutlak değerini döndürür.	i = Abs(-3); // i = 3
Odd()	Verilen sayının tek olup olmadığını kontrol eden bir fonksiyondur. Tek ise true değeri döndürür, çift ise false değeri döndürür.	Odd(5) //True
Random()	0 ve 1 arasında rastgele bir değer üretir.	x = Random(); x = Random() * 100; //0 ve 100 arasında değerler üretir.
Dec()	Integer tipindeki değişkenin değerini bir sayı azaltır.	Dec(sayi); // sayi = 23 => sayi = 22
Inc()	Integer tipindeki değişkenin değerini bir sayı arttırır.	Inc(sayi); // sayi = 23 => sayi = 24
Round()	Ondalıklı sayıları en yakın tam sayıya yuvarlamak için kullanılır. Ondalık kısmı 0.5 üstünde ise bir üst sayıya, 0.5'ten küçük olanları bir alta yuvarlar. Verilen sayı her iki sayının tam ortasındaysa (0.5) sonuç her zaman çift sayıdır.	Sayi = Round(15.50); // Sayi = 16 // 16.50 = 16 // 16.51 = 17 // 16.49 = 16
Div()	İki sayının bölünmesi sonucu tam sayı kısmını döner.	Sayi = 27 Div 2; //Sayi = 13
Mod()	Bir sayının diğer bir sayıya bölümünden kalanı döner.	Sayi = 27 mod 2; //Sayi = 1
Frac()	Ondalıklı sayının ondalık kısmını döndürür.	Sayi = Frac(12.75) //Sayi = 0,75
Trunc()	Ondalıklı sayının tam sayı kısmını döndürür.	Sayi = Trunc(12.75) //Sayi = 12
Sqrt()	Pozitif bir değerin karekökünü bulmak için kullanılır.	Sayi = Sqrt(16); // Sayi = 4

Tablo 12: Matematik Fonksiyonları

Örnek 1

Tablo 12'deki matematik fonksiyonları basit bir şekilde kullanılarak bir sayısal veri işlendi ve her işlem açıklamalı bir şekilde mesaj penceresinde gösterildi *(bkz. Kod 67).*

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum8/Ornek8)

Ana Kod

```
var
  sayi: Integer;
  sonuc: Double;
{
  sayi = 15;

  // 1. Tek sayı kontrolü (Odd)
  if (Odd(sayi))
    ShowMessage('Sayı tektir: ' + IntToStr(sayi)) // Çıktı: 'Sayı tektir: 15'
  else
    ShowMessage('Sayı çifttir: ' + IntToStr(sayi));

  // 2. Sayıyı bir artır (Inc)
  Inc(sayi);
  ShowMessage('Sayı bir artırıldı: ' + IntToStr(sayi)); // 16

  // 3. Yuvarlama (Round)
  sonuc = 15.75;
  sonuc = Round(sonuc);
  ShowMessage('15.75 yuvarlandı: ' + FloatToStr(sonuc)); // 16

  // 4. Bölüm sonucu (Div)
  sonuc = 16 div 3; // Tam sayı bölme
  ShowMessage('16 div 3 sonucu: ' + FloatToStr(sonuc)); // 5

  // 5. Mod işlemi (kalan bulma)
  sonuc = 16 mod 3; // Kalan
  ShowMessage('16 mod 3 sonucu: ' + FloatToStr(sonuc)); // 1

  // 6. Ondalıklı kısmı al (Frac)
  sonuc = Frac(15.75);
  ShowMessage('15.75 ondalık kısmı: ' + FloatToStr(sonuc)); // 0,75

  // 7. Tam kısmı al (Trunc)
  sonuc = Trunc(15.75);
  ShowMessage('15.75 tam kısmı: ' + FloatToStr(sonuc)); // 15
}
```

Kod 67: Matematiksel Fonksiyonlar Örnek 1

Örnek 2

Örnekte, bazı temel matematiksel işlemler gösterilmektedir. İlk olarak, ‘z’ değişkenine 5 değeri atanır ve Dec(z) işlemi ile ‘z’ bir sayı azaltılarak 4 olur. Ardından bu değer ekranda gösterilir. Daha sonra ‘x’ değişkenine 25 değeri atanır ve Sqrt(x) fonksiyonu kullanılarak ‘x’in karekökü ‘y’ değişkenine atanır ve ekranda 5 olarak gösterilir. Ardından, ‘x’ değişkenine -10 değeri atanır ve Abs(x) fonksiyonu kullanılarak ‘x’ in mutlak değeri ‘y’ değişkenine atanır ve ekranda 10 olarak gösterilir. Son olarak, Random fonksiyonu kullanılarak 0 ile 20 arasında rastgele bir sayı üretilir ve bu sayı ekranda gösterilir. Bu kod, azaltma, karekök alma, mutlak değer bulma ve rastgele sayı üretme işlemlerini örnekler *(bkz. Kod 68).*

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum8/Ornek9)

Ana Kod

```
var
  y : Double;
  x,z : Integer;
{
  // Bir sayının bir azaltılması
  z = 5;
  Dec(z);
  ShowMessage('5-1 = '+ IntToStr(z)); //4

  // Karekök alma
  x = 25.0;
  y = Sqrt(x);
  ShowMessage('25''in karekökü: '+ IntToStr(y)); //5

  // Mutlak değer alma
  x = -10;
  y = Abs(x);
  ShowMessage('-10''un mutlak değeri: '+ IntToStr(y)); //10

  // Rastgele sayı üretme
  x = Random()*20;
  ShowMessage('Rastgele sayı: '+ IntToStr(x));
}
```



Kod 68: Matematiksel Fonksiyonlar Örnek 2

ZAMAN VE TARİH FONKSİYONLARI

Zaman ve tarih fonksiyonları, programlama dillerinde tarih ve saat ile ilgili işlemleri gerçekleştirmek için kullanılan araçlardır ve birçok uygulama alanında önemli bir rol oynar. Bu fonksiyonlar; tarih ve saat bilgilerini elde etmek, karşılaştırmak, formatlamak ve manipüle etmek gibi işlemlerin kolay ve hızlı bir şekilde yapılmasına olanak tanır. Günlerden ne olduğu, saatin kaç olduğu ya da günün tarihi bu fonksiyonlarla bir insana sorar gibi bilgisayara sorulabilir.

Tablo 13'de zaman ve tarih fonksiyonları için örnekler görülebilir.

Fonksiyon	Özellik	Kullanımı
Date	Mevcut sistem tarihini döndüren bir fonksiyondur. Bu fonksiyon, yalnızca gün, ay ve yıl bilgilerini döndürür.	GecerliTarih = Date; //08.01.2024
Time	Mevcut sistemdeki zamanı saat, dakika ve saniye türünden döndüren bir fonksiyondur.	GecerliSaat = Time; //16:35:49
Now	Mevcut sistem tarihini ve saatinin tam bilgisini döndürür.	GecerliTarihSaat = Now; //08.01.2024 16:35:49
DayOfWeek()	Belirtilen bir tarihin, haftanın hangi gününe denk geldiğini belirlemek için kullanılır. Haftanın günleri, Pazar'dan Cumartesi'ye kadar 1'den 7'ye kadar olan sayılarla temsil edilir.	Deger = DayOfWeek('08.01.2024'); //2
IncMonth()	Verilen bir tarih değerine belirli bir sayıda ay eklemek için kullanılır. Bu fonksiyon, tarih ve saat bileşenlerini korurken ay bazında artış veya azalış yapar.	TarihDegeri = IncMonth('08.10.2023', 3); //08.01.2024

Tablo 13: Zaman ve Tarih Fonksiyonları

Kod 69'da, 'Tarih' adında bir tarih-saat değişkeni ve 'GunSayisi' adında bir tam sayı değişkeni tanımlandı. İlk olarak, *EncodeDate*(2023,02,27) fonksiyonu kullanılarak 27 Şubat 2023 tarihi 'Tarih' değişkenine atandı ve DayOfWeek(tarih) fonksiyonu ile bu tarihin haftanın hangi günü olduğu hesaplanarak 'GunSayisi' değişkenine atandı. Case yapısı kullanılarak 'GunSayisi' değişkeninin değeri kontrol edildi ve her bir değere karşılık gelen gün adı (Pazar, Pazartesi, Salı, Çarşamba, Perşembe, Cuma, Cumartesi) ShowMessage fonksiyonu ile ekranda gösterildi.

Ardından 'Tarih' değişkenine Now() fonksiyonu ile mevcut tarih, saat atandı ve *DateTimeToStr*(tarih) fonksiyonu kullanılarak bu tarih ve saat string olarak dönüştürüldü, ekranda gösterildi. Bu yapı, belirli bir tarihin haftanın hangi günü olduğunu, mevcut tarih ve saati göstermeyi sağlar.

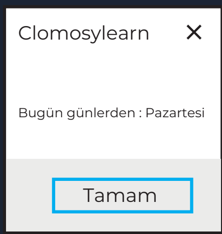
QR Okut

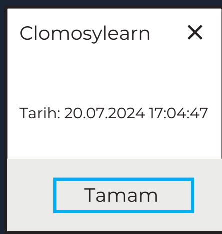
(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum8/Ornek10)

Ana Kod

```
var
 Tarih : TclDateTime;
 GunSayisi   : Integer;
{
 Tarih = EncodeDate(2023,02,27);
 GunSayisi = DayOfWeek(tarih);
 case GunSayisi of
 {
  1: ShowMessage('Bugün günlerden : Pazar');
  2: ShowMessage('Bugün günlerden : Pazartesi');
  3: ShowMessage('Bugün günlerden : Salı');
  4: ShowMessage('Bugün günlerden : Çarşamba');
  5: ShowMessage('Bugün günlerden : Perşembe');
  6: ShowMessage('Bugün günlerden : Cuma');
  7: ShowMessage('Bugün günlerden : Cumartesi');
 }

 Tarih = Now();
 ShowMessage('Tarih : '+DateTimeToStr(Tarih));
}
```





Kod 69: Zaman ve Tarih Fonksiyonları Örneği

Alıştırmalar

1. Bir prosedür yazınız. Bu prosedürün adı RenkSecici olsun. Prosedür, rastgele bir renk (Kırmızı, Mavi, Yeşil, Mor, Sarı, Turuncu, Kahverengi, Pembe) seçerek ekrana şu formatta bir mesaj yazdırınız:

   ```
   "Bugünün rengi: [Renk]"
   ```

2. Bir fonksiyon yazınız. Fonksiyonun adı UzunlukKarsilastir olsun. Bu fonksiyon, iki string parametresi almalı ve uzunluğu daha fazla olan string'i geri döndürmelidir. Eğer iki string'in uzunluğu eşitse, ekrana eşitliği belirten bir geri dönüş sağlayınız.

3. Kullanıcıdan bir kelime alınız ve bu kelimenin her harfi arasına bir boşluk ekleyiniz. Insert fonksiyonunu kullanarak her harf arasına bir boşluk ekleyiniz. Örnek:

   ```
   "Girdi: kedi"
         "Çıktı: k e d i"
   ```

4. Bir parola belirleyiniz ve Insert ve AnsiUpperCase fonksiyonlarını kullanarak aşağıdaki işlemleri yapınız:
   - Parolanın sonuna "!" ekleyiniz.
   - Parolanın tamamını büyük harfe çeviriniz.
   - Sonucu ekrana yazdırınız.

5. Kare ve dik üçgenin kısa ve uzun kenar uzunluğu girilerek alanını hesaplayan iki ayrı prosedür tanımlayınız ve sonuçları gösteriniz.

6. Dairenin alanını ve çevresini hesaplayan ve geriye değer döndüren fonksiyonu yazınız.

https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum8

Ek Kaynaklar

- Void kullanımı
  https://www.docs.clomosy.com/Procedures

- Function kullanımı
  https://www.docs.clomosy.com/Functions

- Zaman ve tarih fonksiyonları kullanımı
  https://www.docs.clomosy.com/System_Library#Date_and_Time_Functions

- Matematik fonksiyon kullanımı
  https://www.docs.clomosy.com/System_Library#Math_Functions

- Karakter işleme fonksiyon kullanımı
  https://www.docs.clomosy.com/System_Library#String_Functionss

BÖLÜM 9 : HATA YAKALAMAK

Uygulama geliştirme sürecinin en önemli aşamalarından biri test sürecidir. Bu aşamada yapılan hata tespit edilerek düzeltilebilir. Hata yakalamak, bir program çalışırken ortaya çıkabilecek hataları kontrol etmek ve uygun bir şekilde tepki vermek için kullanılır.

Hata yakalamanın kullanım alanları:

- Programın daha güvenilir olması
- Kullanıcı dostu geri bildirim
- Programın daha iyi bakımı
- Geliştirme sürecini kolaylaştırma
- Kodun daha verimli olması

TRY-EXCEPT

Yazılım geliştirme sürecinde hatalarla karşılaşmak kaçınılmazdır. Bu hatalar, beklenmedik durumlar, kullanıcı hataları veya sistem arızaları gibi çeşitli nedenlerden kaynaklanabilir. Clomosy'de bu tür hataları yakalamak ve yönetmek için kullanılan temel yapı taşlarından biri *try-except* bloğudur. Try-except, kodunuzun belirli bir bölümünü hatalardan korumanıza ve bu hataları yönetmenize olanak tanır.



Try-except Bloğunun Temel Yapısı:

Bir try-except bloğu, iki ana bölümden oluşur:

- **'try' Bölümü:** Burada, programın normal olarak yapması istenen kod yazılır. Bu bölümdeki kod çalıştırılırken bir hata oluşursa, kodun çalışması durdurulur ve except bölümüne geçilir.
- **'except' Bölümü:** try bloğunda bir hata yakalandığında bu bölüm devreye girer. Program hataya düştüğünde yapılması gereken işlemler burada yazılacaktır. Hata oluştuğunda gösterilecek bir mesaj veya hatanın nasıl düzeltileceğine dair işlemler bu kısımda yer alır.

Try-Except şu şekildedir:

```
try // Hata oluşabilecek kod bloğu
  // Bu blok içinde bir hata oluşursa, program try bloğundan çıkar ve except
bloğuna geçer
except // hata olması durumunda bu bölümde bulunan kod satırları çalışır.
```

BİLGİLENDİRME:
try bloğunda, kod çalışırken bir hata (istisna) oluştuğunda, bu hatanın meydana geldiği satırdan itibaren try bloğunun geri kalan kısmı atlanır ve doğrudan except bloğuna geçilir.

Mesela, bir kahve makinesini kullanılırken her zaman kahve sorunsuzca hazırlanmak istenir. Ancak, makineye su koymayı unutmak gibi beklenmedik durumlar yaşanabilir. Bu durumda, makinenin su yokken çalışmaya devam etmesi yerine durup kullanıcıya uyarı vermesi gerekir. İşte bu durum, yazılımda try-except bloğunun işlevine benzer.

Senaryo:

- **try Bloğu:** Kullanıcı, kahve makinesinin çalıştır düğmesine basar. Makine, suyun olduğunu farzederek işlemleri yaparken hataya düşer.
- **except Bloğu:** Eğer su yoksa makine işlemi durdurur ve "Su eksik, lütfen su ekleyin." şeklinde bir uyarı verir.

Örnek 1

Kod 70’te üç tam sayı değişkeni (x, y ve Sonuc) tanımlanır. x değişkenine 0 ve y değişkenine 15 değeri atanır. Daha sonra, y değişkeninin x değişkenine bölünmesi (y div x) işlemi gerçekleştirilir ve sonuç ‘Sonuc’ değişkenine atanır. Ancak, x değeri 0 olduğu için 0'a bölme hatası meydana gelir ve bu hatanın sonucu olarak program çalışırken bir hata mesajı görüntülenir *(bkz. Görsel 61)*. Son olarak, ShowMessage fonksiyonu ile “Sonuc”değişkeninin değeri bir mesaj kutusunda gösterilmek istenir ancak bu satır hata nedeniyle çalışmayacaktır.

Ana Kod

```
Var
  x,y,Sonuc : Integer;
{
    x = 0;
    y = 15;
    Sonuc = y div x;
    ShowMessage(IntToStr(Sonuc));
}
```

Kod 70: Try – Except Hata Örneği

Console

C TRObject Language Main

```
Var
   x,y,sonuc : Integer;
{
x=0;
y=15;
sonuc=y div x;
ShowMessage(IntToStr(sonuc));
}

----------------------------------------------------------------------------------------------------
RUNTIME ERROR
File library Main: Division by zero when evaluating instruction OperDiv ($0,$0,$0,$0,' ').
Stack content is: [Null,15,0].
Sounce position: 6,16
Position: 6,16
Clomosy Build: 20241204.102408
----------------------------------------------------------------------------------------------------
```

Görsel 61: Try – Except Konsol Hata Mesajı

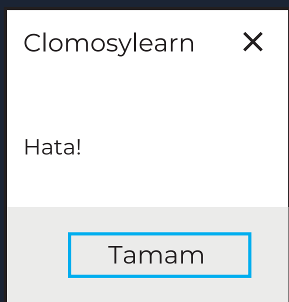
Aynı örnek try-except blokları içerisine alındığında meydana gelen 0'a bölme hatası except bloğu tarafından yakalanır. Hata oluştuğunda 'Sonuc' değişkeninin değeri yerine 'Hata!' mesajı gösterilir *(bkz. Kod 71)*. Bu sayede, program hata aldığında çalışmayı durdurmaz ve kullanıcıya uygun bir hata mesajı gösterir.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum9/Ornek1)

Ana Kod

```
Var
 x,y,Sonuc : Integer;
{
 x = 0;
 y = 15;
 try
   Sonuc = y div x;
   ShowMessage(IntToStr(Sonuc));
 except
   ShowMessage('Hata!');
 }
}
```



Kod 71: Try – Except Örneği

Örnek 2

Kod 72'de, yanlış tür dönüşümü yapılmasından kaynaklı meydana gelen hata ele alındı.

Ana Kod

```
Var
 Deger : String;
{
   Deger = '15b16';
   ShowMessage(StrToInt(Deger));
}
```

Kod 72: Tür Dönüşümü Hata Örneği

Örnekte, 'Deger' adında bir string değişkeni tanımlandı ve bu değişkene '15b16' değeri atandı. Ardından, StrToInt fonksiyonu ile 'Deger' değişkenindeki string değer bir integer türüne dönüştürüldü. Ancak '15b16' değeri geçersiz bir tam sayı formatı içerdiğinden, StrToInt fonksiyonu dönüşüm sırasında bir hata meydana getirdi ve program hata mesajı gösterdi *(bkz. Görsel 62).*

Hata Mesajı;

Console

C TRObject Language Main

```
Var
   deger : String;
{
   deger = '15b16'
   ShowMessage(StrToInt(deger));
}
-----------------------------------------------------------------------------------------------
RUNTIME ERROR
File library Main: '15b16'  is not a valid integer value when evaluating instruction CallProc ($1,$1,$165A098,$197b7268, 'StrToInt').
Stack content is: [UnicodeString:15b16].
Sounce position: 5,29
Position: 5,29
Clomosy Build: 20241204.102408
-----------------------------------------------------------------------------------------------
```

Görsel 62: Örnek 2 Konsol Hata Mesajı

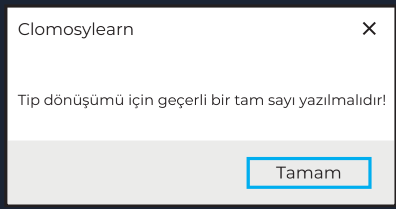
Şimdi kodu düzenleyelim.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum9/Ornek2)

Ana Kod

```
Var
 Deger : String;
{
 Deger = '15b16';
 try
   ShowMessage(StrToInt(Deger));
 except
   ShowMessage('Tip dönüşümü için geçerli bir tam sayı değeri yazılmalıdır!');
 }
}
```



Kod 73: Try – Except / Tür Dönüşümü Kontrolü

Try bloğu içinde, StrToInt fonksiyonu ile 'Deger' değişkenindeki string değer bir integer türüne dönüştürüldü. Oluşacak hata except bloğu tarafından yakalandı ve kullanıcıya "Tip dönüşümü için geçerli bir tam sayı değeri yazılmalıdır!" mesajı gösterildi *(bkz. Kod 73)*. Bu sayede, geçersiz veri nedeniyle oluşan hata yönetildi ve programın durması engellendi.

TRY-FINALLY

Yazılım geliştirme sürecinde, sadece hataları yakalamak değil, aynı zamanda kullanılan kaynakları doğru bir şekilde yönetmek de kritik bir öneme sahiptir. Clomosy'de, bu tür durumları yönetmek için kullanılan yapılardan biri *try-finally* bloğudur.

Try-finally, belirli bir kod bloğunun her durumda çalışmasını garanti eder bu da özellikle kaynakları bellekte serbest bırakma gibi kritik işlemler için oldukça önemlidir.



Try-finally Bloğunun Temel Yapısı:

Bir try-finally bloğu iki ana bölümden oluşur:

- **Try Bölümü:** Hata oluşmasına neden olabilecek kod bu bölümde yer alır. Kod çalıştırıldıktan sonra hata oluşup oluşmadığına bakılmaksızın finally bloğuna geçer.
- **Finally Bölümü:** Bu bölüm, her durumda çalıştırılması gereken kodları içerir. try bloğunda bir hata meydana gelse bile finally bloğundaki kod mutlaka çalışır. Bu, kaynakların serbest bırakılması veya kritik işlemlerin tamamlanması gerektiğinde önemlidir. Ancak Finally bölümünde bir hata olursa devamındaki kodlar çalışmaz.

Try-finally yapısı şu şekildedir:

```
try
  // Bu bölümde bulunan kod satırları, normal akışta çalıştırılır. Eğer bu bölümde bir hata
 // oluşursa hata yönetimi yapılabilir ya da kodun çalışması beklenmedik bir şekilde durabilir.
finally
  //Bu bölümde bulunan 'try' bloğunda bir hata oluşsa bile mutlaka çalıştırılacaktır.
}
```

Örneğin, evden çıkarken kapıyı kilitlemek için yapılan bir işlemi ele alalım. Kişi, evden çıkmadan önce cüzdanını, anahtarlarını ve diğer gerekli eşyalarını almayı planlar. Ancak, bu işlemler sırasında bir şeyler unutulabilir veya beklenmedik bir durum oluşabilir. Her ne olursa olsun, kişi evden çıkarken kapıyı kilitlemek zorundadır.

Senaryo:

- **try Bölümü:** Evden çıkmadan önce yapılacak işler; cüzdanı almak, ışıkları kapatmak, eşyaları toplamak vb.
- **finally Bölümü:** Her durumda kapıyı kilitlemek.

Örnek

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum9/Ornek3)

Ana Kod

```
Var
   Sayi, Sayi2,Toplam : Integer;
{
   Sayi = 20;
   Sayi2 = 18;
 try
   Toplam = Sayi + Sayi2;
 finally
   ShowMessage('Toplama işlemi gerçekleştirildi.');
   ShowMessage(IntToStr(Sayi)+' + '+IntToStr(Sayi2)+' = '+IntToStr(Toplam));
 }
}
```

Kod 74: Try – Finally Örneği

Kod 74'te, 'Sayi' ve 'Sayi2' adında iki tam sayı değişkeni tanımlandı ve sırasıyla 20 ve 18 değerleri atandı. 'Sayi' ve 'Sayi2' değişkenlerinin toplamı try bloğu içerisinde 'Toplam' değişkenine atandı. finally bloğu, try bloğunda hata olup olmadığına bakmaksızın her durumda çalışacak kodu içerdiği için iki mesaj gösterdi: İlki "Toplama işlemi gerçekleştirildi." mesajıdır, ikincisi ise 'Sayi' ve 'Sayi2' değişkenlerinin değerleri ve toplamlarını gösteren "IntToStr(Sayi)+' + '+IntToStr(Sayi2)+' = '+IntToStr(Toplam)" ifadesidir. Bu yapı, toplama işleminin sonucunu her durumda kullanıcıya bildirmek için kullanılır.

Alıştırmalar

1. Kullanıcıdan tarih girdisi alıp bu tarihi *DateTime* formatına dönüştürünüz. Dönüştürme sırasında bir hata oluşursa kullanıcıya hata mesajı gösteriniz.

2. Uygulamada bir sayı değişkeni tanımlayarak bu sayının karekökünü hesaplayan bir program yazınız. Eğer girilen sayı negatifse bir hata mesajı gösteriniz.

3. Kullanıcının girdiği iki sayıyı toplayınız ve eğer geçersiz bir giriş yapılırsa uygun bir hata mesajı gösteriniz.

https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum9

Ek Kaynaklar

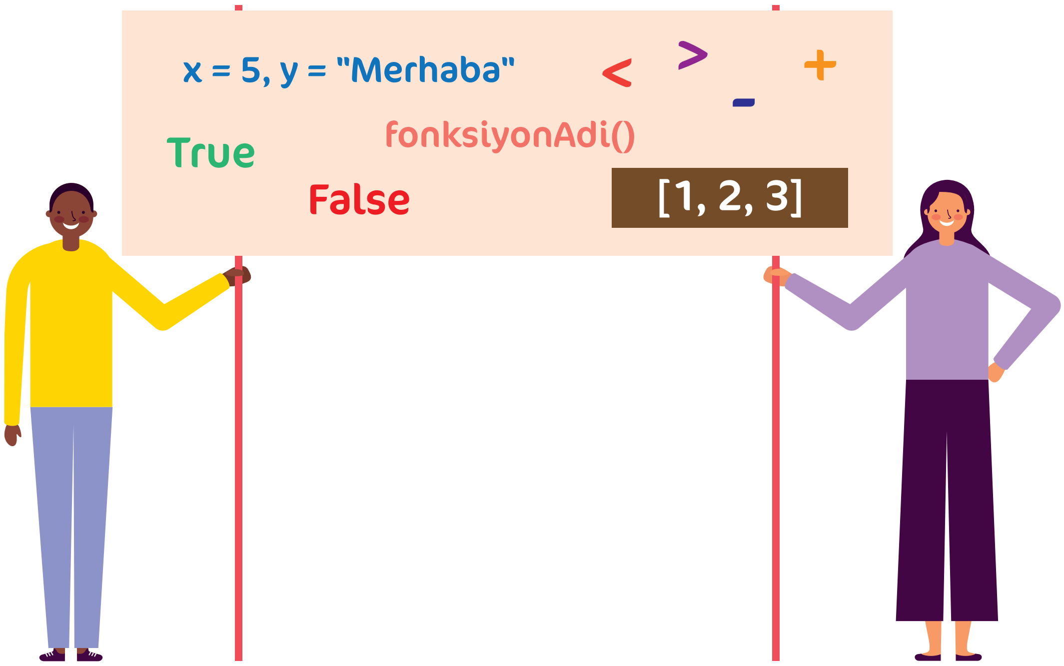
- Try-Except Kullanımı
  https://www.docs.clomosy.com/Error_Trapping#Try...finally_Blocks

- Try-Finally Kullanımı
  https://www.docs.clomosy.com/Error_Trapping#Try...except_Blocks

BÖLÜM 10 : FORMLAR

FORM NEDİR?

Bu bölüme kadar değişkenler, operatörler, koşul yapıları, döngüler, fonksiyonlar ve diziler gibi programlama temelleri ele alındı. Bu yapılar, geliştirilen programların arka planda çalışan ve son kullanıcı tarafından görülmeyen bileşenlerdir.



Program arayüz tasarım araçları: Formlar, Metin kutusu ve butonlar. Programın kullanıcıya görünen kısmı hem işlevsel olmalıdır hem de bir sanatçının elinden çıkmışçasına göze hoş görünmelidir. Öyleyse bir ressamın boş bir tuvalle başladığı gibi öncelikle tasarım ögelerinin yerleştirileceği bir alana ihtiyaç vardır. Clomosy'de bu görevi formlar üstlenir.

Formlar; resim tuvali, yazı tahtası veya boş bir kâğıt gibi düşünülebilir. Ona istenilen renk verebilir ya da bir görsel form üzerine yerleştirilebilir. Ayrıca form, klavye ve tıklamalar aracılığıyla kullanıcının etkileşimlerine yanıt verebilir.

Clomosy'de çeşitli görevler için özelleştirilmiş 6 form bulunmaktadır:

- TclForm
- TclGameForm
- TclDrawForm
- TclStyleForm
- TclGuideForm
- TclSyntaxForm

Bu bölümde diğer formların da temelini oluşturan *TclForm* nesnesi ele alınacaktır.

TCLFORM

Öncelikle 'AnaForm' adı verilen bir *TclForm* nesnesi tanımlandı.

```
var
  AnaForm:TclForm;
```

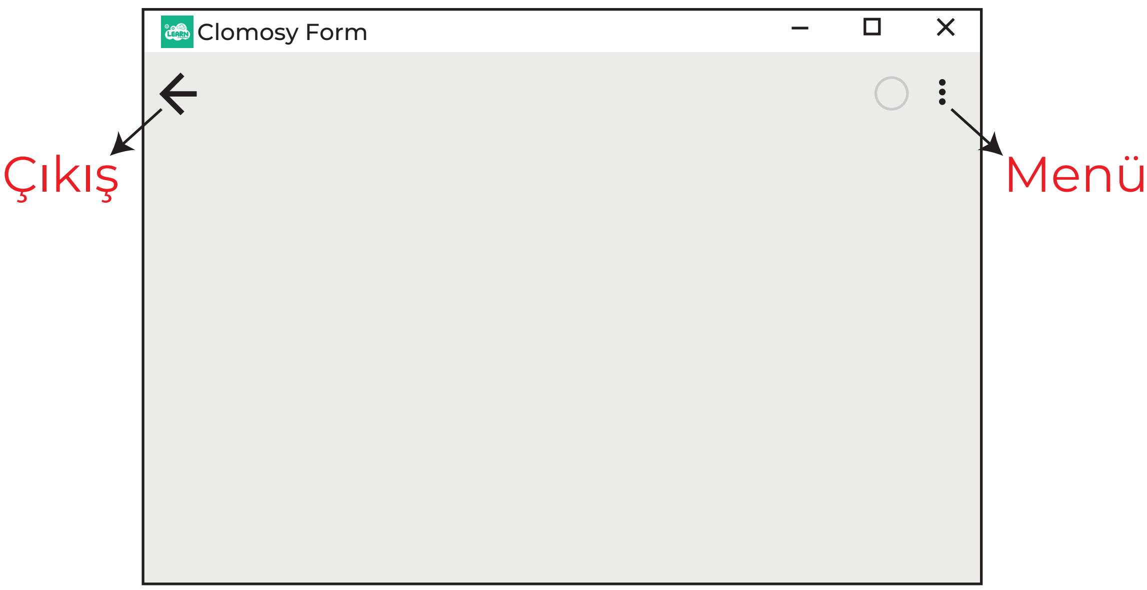
Şimdi ana kod bloğunda *Create* özelliği kullanılarak form oluşturuldu ve *Run* özelliği kullanılarak form görünür hâle getirildi *(bkz. Kod 75)*. Bu noktada Run özelliğini kullanmayı atlamamak önemlidir. Run özelliği bütün sistemi kurduktan sonra güç düğmesine dokunup bilgisayarı başlatmak gibidir. Eğer Run özelliği işleme alınmazsa form görünemez.

Ana Kod

```
var
 AnaForm:TclForm;
{
 AnaForm=TclForm.Create(Self);
 AnaForm.Run;
}
```

Kod 75: TclForm Kullanımı

Kod kaydedilip çalıştırıldığında *Görsel 63*'teki form ekrana gelecektir.



Görsel 63: TclForm Ekranı

Evet, artık forum kullanılmaya hazır. Form boş bir tuvale benzetilmişti. Ancak aslında Clomosy'de oluşturulan formlar bundan çok daha fazlasını sunar. Form oluşturulduğunda form üzerinde standart olarak tanımlanmış olan çıkış ve menü butonları formun üst köşelerinde yer alır. İsteğe göre bu butonların ekranda görünmemesi sağlanabilir.

Bu işlemi yapabilmek için form üzerindeki standart butonların isimlerine göre *Visible* değerlerini False yapmak gerekir. Geri butonu bileşeninin ismi 'BtnGoBack' ve menü butonu 'BtnFormMenu' olarak geçer. Bunların Visible değerleri False olduğunda ekranda görünmemektedir. Kullanımı:

```
AnaForm.BtnFormMenu.Visible = False;
AnaForm.BtnGoBack.Visible = False;
```

Clomosy'de formlar çeşitli form özellikleri kullanılarak özelleştirilir.

FORM ÖZELLİKLERİ

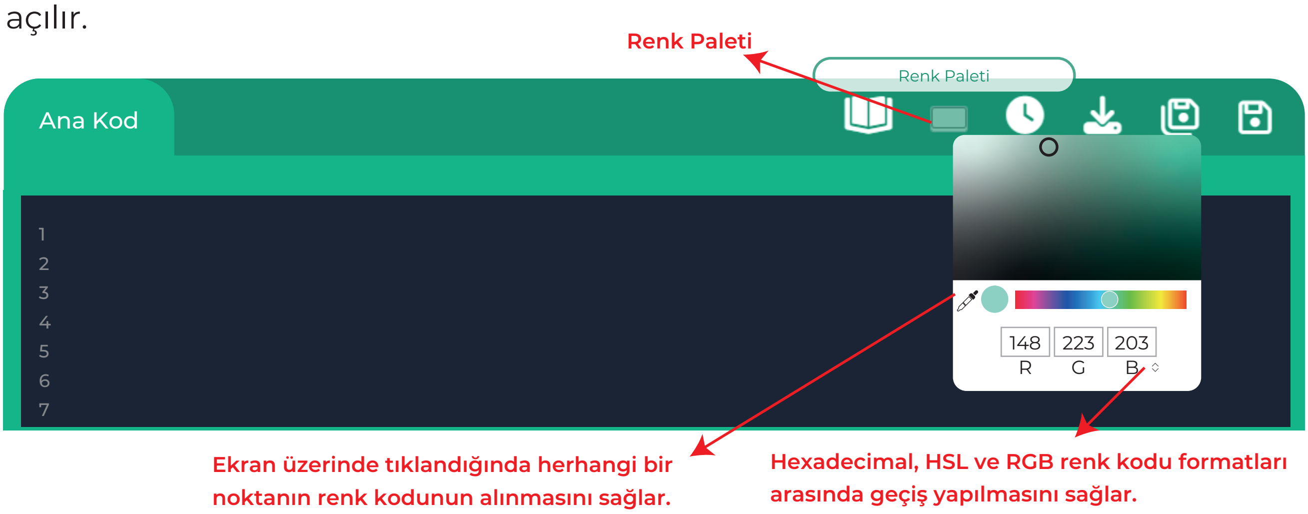
ARKA PLANA RENK VERME

SetFormColor özelliği kullanılarak formun arka planı renklendirilir. Metod kullanılırken üç parametre belirtilmesi gerekir. İlk iki parametre, kullanılacak renklerin hexadecimal kodlarıdır. Son parametre ise renk geçişi ayarlamak içindir. Yani arka planda tek renk ya da renk geçişi olması bu değer ile ayarlanır. Clomosy'de 4 farklı renk geçiş yöntemi kullanımı bulunmaktadır *(bkz. Tablo 14).*

Özellik	Kullanımı
clGNone	Tek renk kullanımı içindir.
clGVertical	Dikey renk geçişi oluşturur.
clGHorizontal	Yatay renk geçişi oluşturur.
clGCross	Çapraz bir renk geçişi oluşturur.

Tablo 14: Form Arka Plan Renk Özellikleri

Clomosy geliştirici ortamında, renk kodlarının kolayca alınabileceği bir buton bulunmaktadır. Görseldeki işaretli kısma tıklandığında Clomosy renk paleti açılır.



Clomosy renk paletini kullanarak ekran üzerine tıklandığında; bir noktanın renk kodu ya da renk paleti üzerinden seçilen bir rengin kodu, Hexadecimal, HSL veya RGB formatında öğrenilebilir.

Arka planda tek renk olması için özelliğin kullanımı *Kod 76*'daki gibidir. Bu durumda sadece bir renk kodu vermek yeterlidir *(bkz. Görsel 64).*

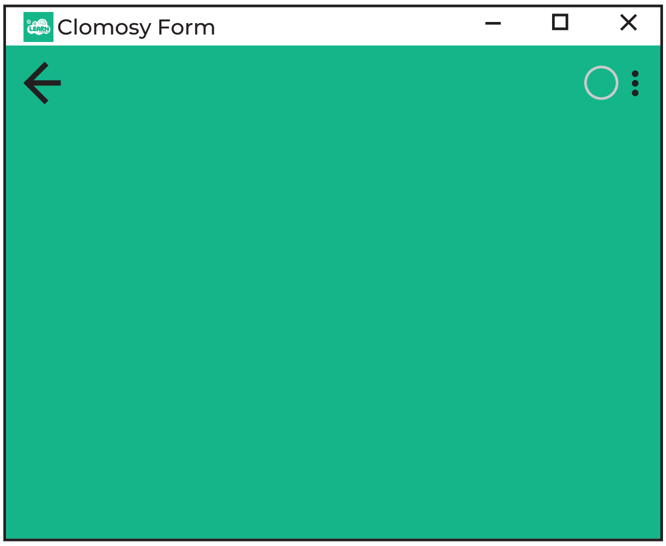
QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum10/Ornek1)

Ana Kod

```
var
  AnaForm:TclForm;
{
  AnaForm=TclForm.Create(Self);
  AnaForm.SetFormColor('#00b282','',clGNone);
  AnaForm.Run;
}
```

Kod 76: TclForm – SetFormColor Örneği 1



Görsel 64: Arka Plan clGNone Kullanımı

Dikey renk geçişli bir arka plan için aşağıdaki gibi *clGVertical* özelliği kullanılır. Bu kullanımda bir renkten diğer bir renge doğru geçiş olacağı için iki rengin de hexadecimal kodunu vermek gerekir.

QR Okut

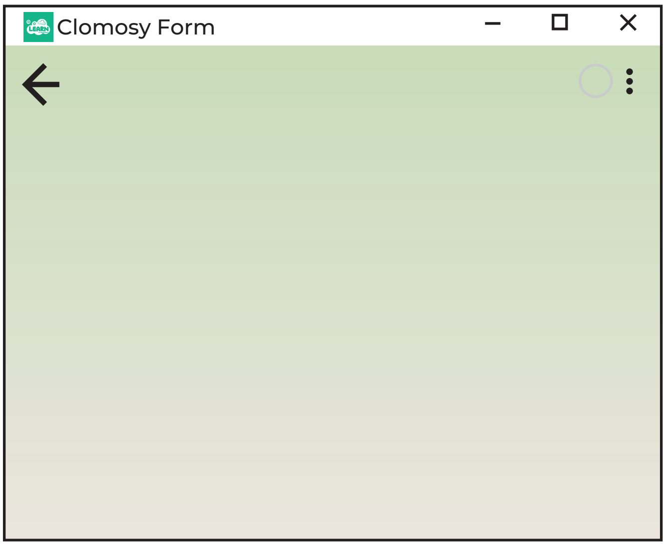
(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum10/Ornek2)

Ana Kod

```
var
  AnaForm:TclForm;
{
  AnaForm=TclForm.Create(Self);
  AnaForm.SetFormColor('#d1dfbb ','#f0ebe5', clGVertical);
  AnaForm.Run;
}
```

Kod 77: TclForm – SetFormColor Örneği 2

Kod 77 kaydedilip çalıştırıldığında aşağıdaki ekran gelecektir *(bkz. Görsel 65).*



Görsel 65: Arka Plan clGVertical Kullanımı

ARKA PLANA GÖRSEL EKLEME

SetFormBGImage özelliği, forma arka plan görseli eklemek için kullanılmaktadır. Özelliği kullanırken görselin URL adresinin string veri türünde belirtilmesi gerekir *(bkz. Kod 78).*

QR Okut

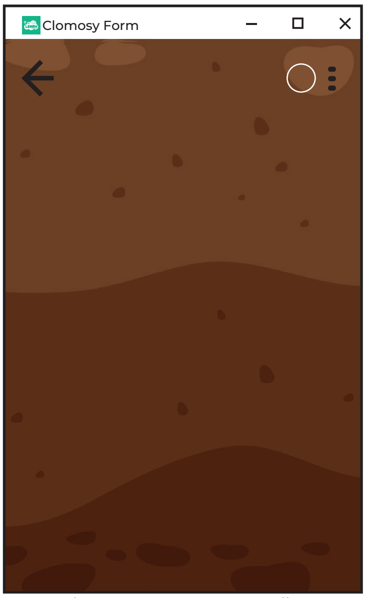
(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum10/Ornek3)

Ana Kod

```
var
  AnaForm:TclForm;
{
  AnaForm=TclForm.Create(Self);
  AnaForm.SetFormBGImage('https://clomosy.com/demos/toprak.png');
  AnaForm.Run;
}
```

Kod 78: TclForm – SetFormBGImage Örneği 1

Kod 78 kaydedilip çalıştırıldığında *Görsel 66* ekrana gelecektir.



Görsel 66: SetFormBGImage Kullanımı

DİKKAT!
Dikey bir görsel kullanılmadığında, görselin ekranda bozulmuş veya genişlemiş şekilde görüntülenmesi mümkündür. Bu nedenle, görsel seçimine dikkat edilmelidir.

Forma görsel eklemenin bir yolu daha var. Bu yolun sadece Windows üzerinde kullanılabileceği unutulmamalıdır. Kullanılmak istenilen görsel, uygulamanın dosya yoluna eklenirse bu dosya içerisinden alınıp uygulamada kullanılabilir. Nasıl yapılacağına gelmeden önce uygulamanın ana dizin ve dosya yolundan bahsedildi.

AppBasePath özelliği Clomosy uygulamasının ana dizin yolunu temsil eden bir özelliktir. *Kod 79* kullanılarak Clomosy uygulamasının yüklü olduğu dizin öğrenilir.

QR Okut

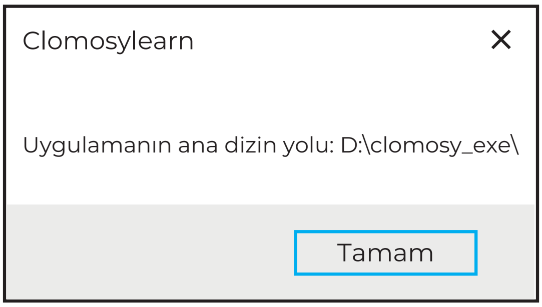
(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum10/Ornek4)

Ana Kod

```
var
 UygulamaYolu : String;
{
 UygulamaYolu = Clomosy.AppBasePath;
 ShowMessage('Uygulamanın ana dizin yolu: '+UygulamaYolu);
}
```

Kod 79: AppBasePath Örneği

Program çalıştırıldığında *Görsel 67* ekrana gelecektir. Sonuç, Clomosy uygulamasının cihazda bulunduğu konuma göre değişir.



Görsel 67: AppBasePath Özelliği Kullanımı

AppFilesPath özelliği uygulamanın dosya yolunu temsil eder. *Kod 80*'i kullanarak uygulamanın dosya yolu öğrenilir.

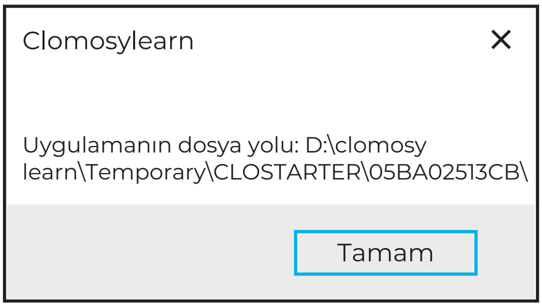
QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum10/Ornek5)

Ana Kod

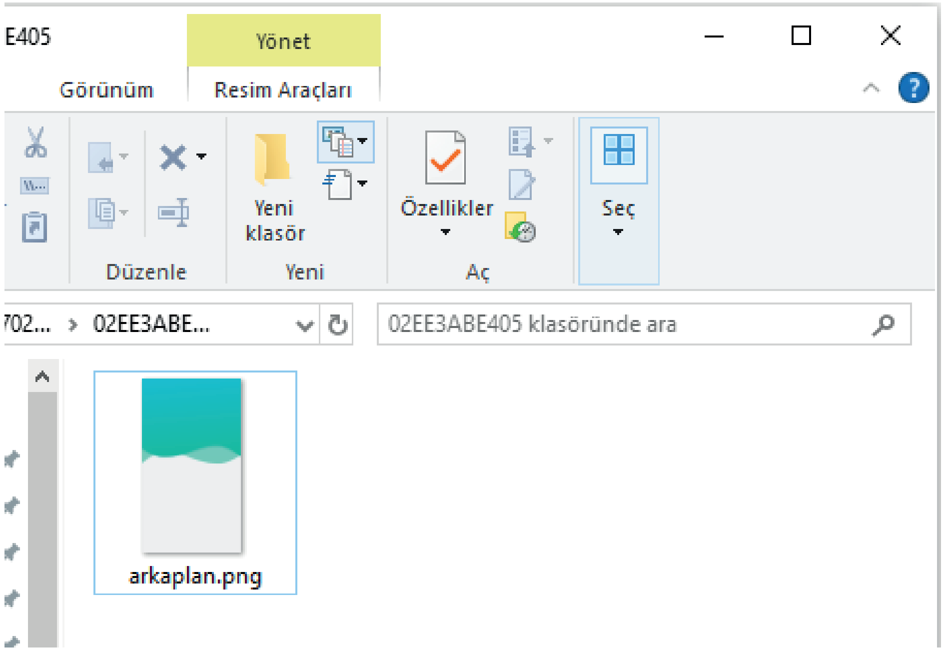
```
var
 UygulamaDosyaYolu : String;
{
 UygulamaDosyaYolu = Clomosy.AppFilesPath;
 ShowMessage('Uygulamanın dosya yolu: '+ UygulamaDosyaYolu);
}
```

Kod 80: AppFilesPath Örneği



Görsel 68: AppFilesPath Özelliği Kullanımı

Bu konuma eklenen bir görsel, uygulamada kullanılabilir *(bkz. Görsel 68)*.



Görsel 69: Dosya Yolu Görüntüleme

Formun arka planına sadece resmin dosya adı yazılarak uygulama çalıştırıldı *(bkz. Kod 81).*

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum10/Ornek6)

Ana Kod

```
var
  AnaForm:TclForm;
{
  AnaForm=TclForm.Create(Self);
  AnaForm.SetFormBGImage('arkaplan.png');
  AnaForm.Run;
}
```

Kod 81: TclForm – SetFormBGImage Örneği 2

Clomosy proje dosyasının bulunduğu dizine istenilen URL adresinden resim nasıl indirilir.

İnternet adresinden dosyayı indirebilmek için *AddAssetFromUrl* fonksiyonu kullanılmalıdır. Bunun için aşağıdaki gibi bir komut yazılmalıdır.

```
AnaForm.AddAssetFromUrl('https://clomosy.com/demos/toprak.png');
```

Bu komut proje dizinine adresdeki dosyayı indirir ve kullanılabilir hâle getirir.

BİLGİLENDİRME:
AddAssetFromUrl fonksiyonu uygulamanın en başında bir kez çalıştırılmalıdır. Görsel adı ile istenilen her yerde kullanılabilir.

AddAssetFromUrl fonksiyonu, geliştirilen uygulamaya belirli bir URL'den dinamik olarak varlık (resim, ses dosyası vb.) eklenmesini sağlar. Bu fonksiyon, özellikle dış kaynaklı varlıkları uygulamaya dâhil etmek için kullanışlıdır. Uygulamayı hızlandırarak kullanıcı deneyimini akıcı hâle getirir.

Örneğin, bir sunucudan doğrudan resimler yüklenebilir ve uygulamada kullanılabilir. Bu, özellikle uygulama varlıklarının güncel tutulması ve uygulama dosyasının boyutunun küçük tutulması açısından faydalıdır.

Clomosy'de kaynakları etkili bir şekilde kullanmak için genellikle bu tür dosyalar çalışma zamanında yüklenir. Bu yöntem, dış varlıkları yönetmeyi kolaylaştırır ve uygulama içinde bunlara kolay erişim sağlar.

İnternet dosyası, proje dizinine kaydedildiğine göre dosya adının yazılması ve aşağıdaki gibi çağırılması yeterlidir.

```
AnaForm.SetFormBGImage('toprak.png');
```

Kod 82'de bu kullanım ile formun arka planına resim eklenmiştir.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum10/Ornek7)

Ana Kod

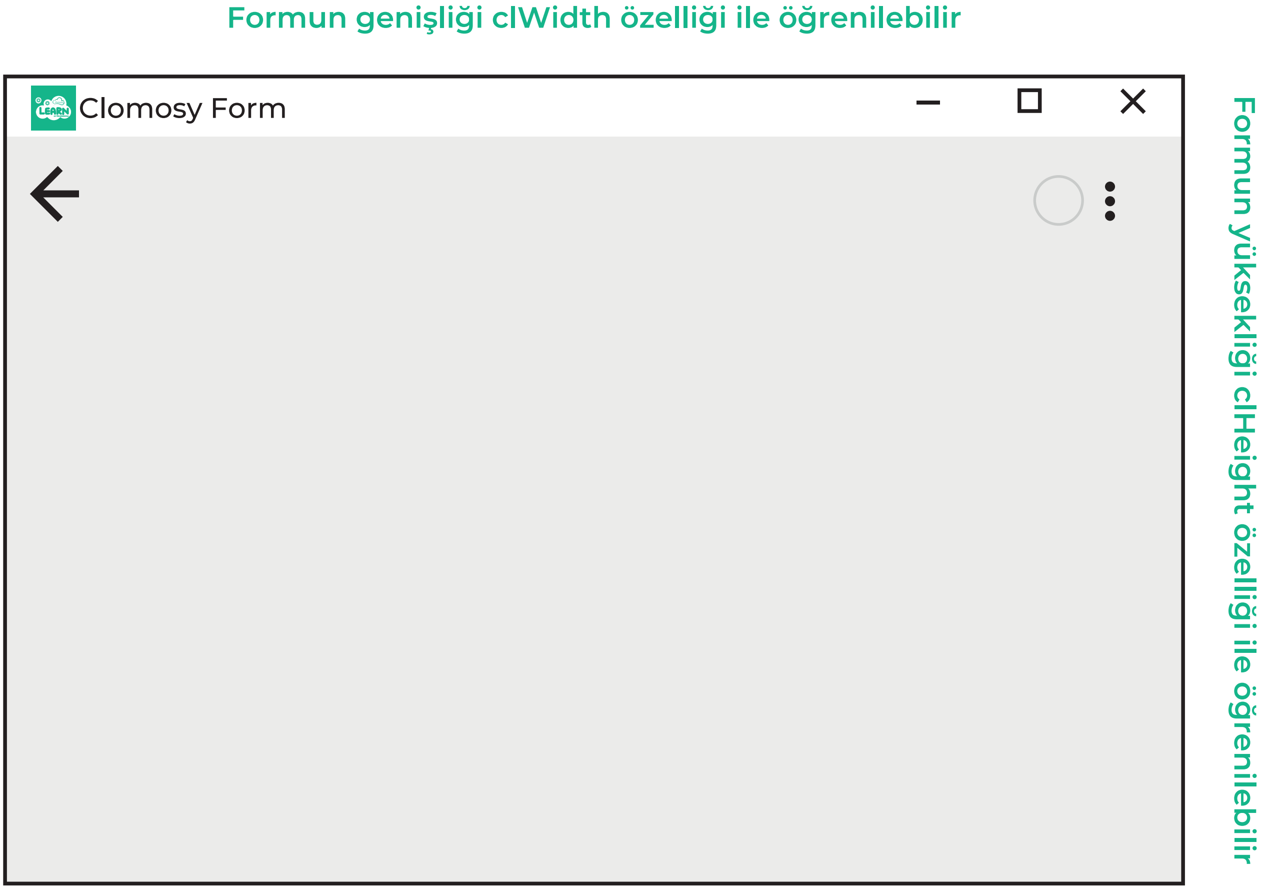
```
var
 AnaForm:TclForm;
{
 AnaForm=TclForm.Create(Self);
 AnaForm.AddAssetFromUrl('https://clomosy.com/demos/toprak.png');
 AnaForm.SetFormBGImage('toprak.png');
 AnaForm.Run;
}
```

Kod 82: TclForm – AddAssetFromUrl Örneği

Kod 82'de AddAssetFromUrl özelliği kullanılarak belirtilen URL adresindeki dosya, proje dosya yoluna kaydedilir. Daha sonra SetFormBGImage özelliğine indirilmiş dosya ismi yazılarak formun arka planına yerleştirilir.

FORM YÜKSEKLİĞİ VE GENİŞLİĞİ

Oluşturulan form program çalıştığında üzerinde çalıştığı mobil işletim sistemi ise cihazın boyutlarını alır. Uygulama geliştirirken tasarım ögelerinin boyutlarının, uygulamanın açıldığı cihaza göre değişmesi istenebilir. Bu durumda form boyutlarının bilinmesi gerekmektedir. Formun genişliğini öğrenmek için *clWidth* ve yüksekliğini öğrenmek için *clHeight* özellikleri kullanılır.



Görsel 70: Form clWidth ve clHeight Özelliği

QR Okut

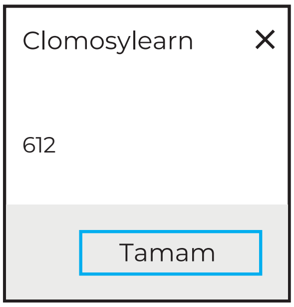
(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum10/Ornek8)

Ana Kod

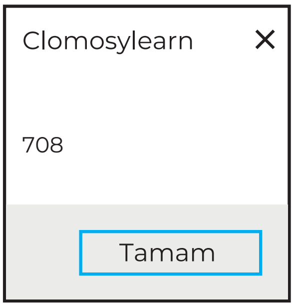
```
var
 AnaForm:TclForm;
{
 AnaForm=TclForm.Create(Self);
 ShowMessage(AnaForm.clHeight);
 ShowMessage(AnaForm.clWidth);
 AnaForm.Run;
}
```

Kod 83: TclForm – clHeight / clWidth Örneği

Program çalıştırıldığında sırasıyla formun yükseklik ve genişlik değerlerini *Görsel 71* ve *Görsel 72*'deki gibi ekrana yazdıracaktır. Kod çalıştırıldığında görülen sayılar farklı olabilir. Çünkü form açıldığı cihaza göre boyut alır.



Görsel 71: Form Yükseklik Değeri



Görsel 72: Form Genişlik Değeri

PENCERE DURUMU AYARLAMA

clSetWindowState özelliği kullanılarak formun Windows üzerinde görüntülenme durumu, formun açılışı veya çalışma zamanı belirlenir. Özelliğin üç farklı kullanım şekli *Tablo 15*'te mevcuttur.

Özellik	İşlevi
fwsMinimized	Form penceresini simge durumuna küçültür.
fwsNormal	Form penceresini normal pencere görünümü yapar.
fwsMaximized	Form penceresini tam ekran durumuna getirir.

Tablo 15: clSetWindowState Özelliği

Kod 84 çalıştırıldığında, form Windows üzerinde tam ekran olarak açılacaktır.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum10/Ornek9)

Ana Kod

```
var
 AnaForm:TclForm;
{
 AnaForm=TclForm.Create(Self);
 AnaForm.clSetWindowState(fwsMaximized);
 AnaForm.Run;
}
```

Kod 84: TclForm – clSetWindowState Örneği

FORM BAŞLIĞINI DEĞİŞTİRME

clSetCaption özelliği kullanılarak form başlığı değiştirilebilir. Özellik kullanılırken formun yeni başlığı string olarak belirtilmelidir. Bu özelliği test edebilmek için Clomosy, Windows işletim sistemi üzerinde açılmalıdır.



Kod 85, form başlığını “Yeni Başlık” olarak değiştirir.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum10/Ornek10)

Ana Kod

```
var
  AnaForm:TclForm;
{
  AnaForm=TclForm.Create(Self);
  AnaForm.clSetCaption('Yeni Başlık');
  AnaForm.Run;
}
```

Kod 85: TclForm – clSetCaption Örneği

Kod bloğu kaydedilip çalıştırıldığında *Görsel 73* ekrana gelir.



Görsel 73: ClSetCaption Özelliği Kullanımı

Alıştırmalar

1. Bir formun arka plan rengini, verilen RGB kodu (111, 2, 120)'yu Clomosy Geliştirme Ortamı'nda "Renk Paleti" üzerinden Hexadecimal koda dönüştürerek değiştiriniz.

2. Windows platformu üzerinde bir form açıldığında aşağıdaki özellikleri içermelidir:
   - Form açıldığında tam ekran olarak ekrana getiriniz.
   - Form başlığını "Hoş Geldiniz" olarak ayarlayınız.
   - Formdaki "Geri" (BtnGoBack) ve "Menü" (BtnFormMenu) butonlarını gizleyiniz.

3. Formun arka planına, kullanıcıdan tuttuğu takımın renklerini belirleyerek dikey bir renk geçişi uygulayan bir program yazınız. Örneğin, Fenerbahçe için sarı ve lacivert, Beşiktaş için siyah ve beyaz kullanılabilir.

https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum10

Ek Kaynaklar

- TclForm Kullanımı
  https://www.docs.clomosy.com/TclForm
  https://www.youtube.com/watch?v=wxWisDIWaxA

- TclStyleForm Kullanımı
  https://www.docs.clomosy.com/TclStyleForm
  https://www.youtube.com/watch?v=16qsR4HvZL8

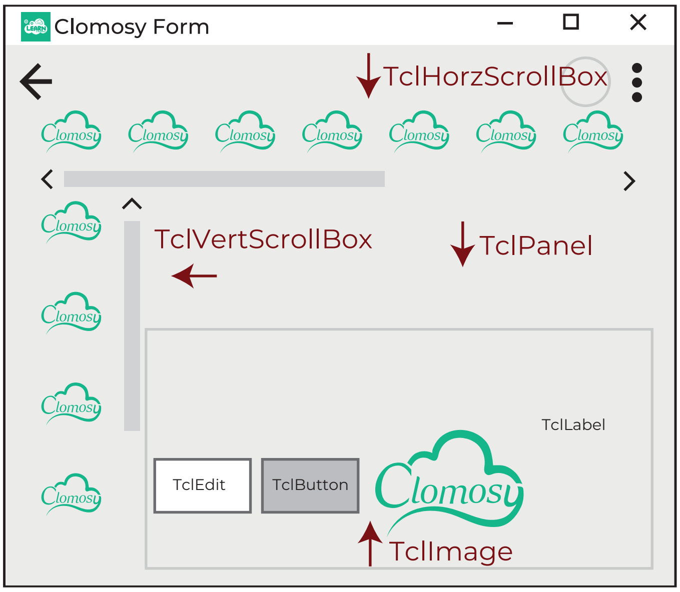
- TclDrawForm Kullanımı
  https://www.docs.clomosy.com/TclDrawForm
  https://www.youtube.com/watch?v=twDasyzM7sU

- TclGuideForm Kullanımı
  https://www.docs.clomosy.com/TclGuideForm
  https://www.youtube.com/watch?v=uXVxu7WQtkQ

- TclGameForm Kullanımı
  https://www.docs.clomosy.com/TclGameForm
  https://www.youtube.com/watch?v=AgxNj6PaJMk&t=7s

BÖLÜM 11 : BİLEŞENLER

Bir önceki bölümde formun nasıl kullanıldığı öğrenildi. Şimdi, form üzerine arayüz tasarımı yapabilmek için bileşenler tanımlanmalı ve tasarlanmalıdır. Bu bölüm, bileşenlerin neler olduğunu, bileşenlerin yerleştirilmesi ve diğer özelliklerinin ayarlanmasını içerir. Böylece, kullanıcı dostu ve işlevsel bir arayüz oluşturulabilir.



Kullanıcı arayüzlerinin tasarımında veya işlevsel ihtiyaçlarda kullanılan birçok özelliği, metodu veya tipi içeren ve tasarım aşamasında oluşum hiyerarşisini gösteren nesnelere *bileşen* denilmektedir.Bileşenler, “component” veya “nesne” olarak da adlandırılmaktadır.

Uygulama çalışma anında, bileşenler görünürlük durumlarına göre “görünür” ve “görünmeyen” olmak üzere iki gruba ayrılmaktadır. Görünür bileşenler, uygulama çalışma anında kullanıcılar tarafından görünen; arka planda çalışan bileşenler ise görünmeyen bileşenlerdir.

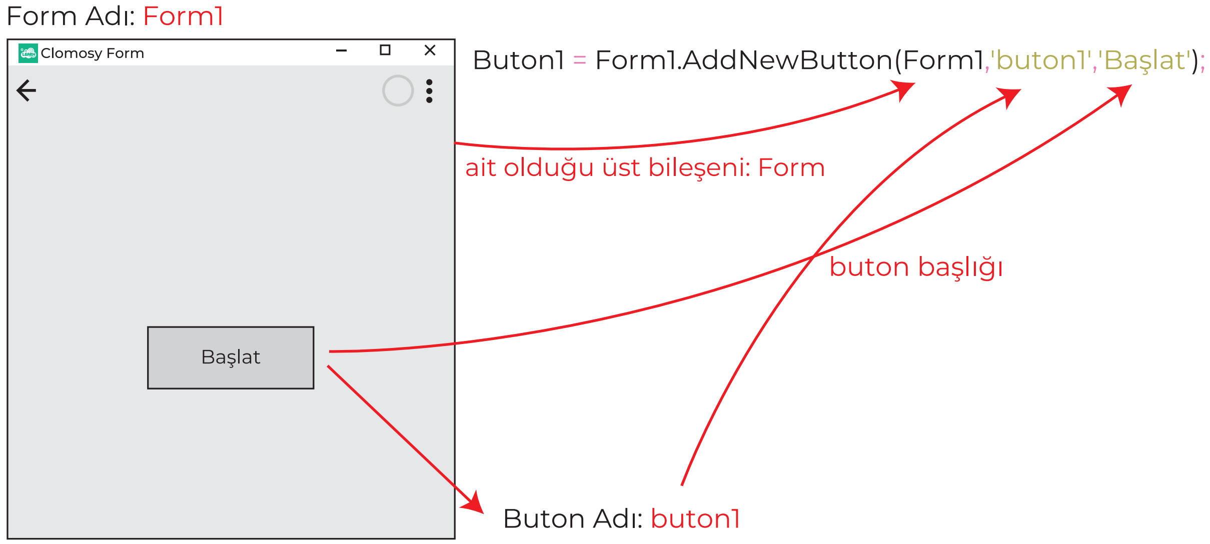
Bunlara örnek olarak görünür bileşenler için *TclButton*, görünmeyen bileşenler için ise *TclTimer* bileşenleri örnek verilebilir. Bu bileşenlere ilerleyen bölümlerde değinilecektir.

TCLBUTTON

Clomosy'de buton, kullanıcı arayüzünde tıklandığında belirli bir eylemi gerçekleştiren bir bileşendir. Bir buton, form üzerine yerleştirildiğinde, kullanıcının tıklayarak belirli işlevi başlatmasını sağlar. Örneğin, "Kaydet" butonu eklenerek ekrandan girilen verilerin kaydedilmesi sağlanabilir.

TclButton, uygulamalarda kullanılmak üzere genel amaçlı bir butondur. Butona tıklandığında bir işlev, prosedür veya diğer eylemler gerçekleşir. Bir butonu oluşturmak için *AddNewButton* metodu kullanılır.

Bu işlev üç parametre alır. İlk parametre, tanımlanacak olan butonun ait olduğu üst bileşeni (parent) belirler. İkinci parametre, tanımlanan butonun adını temsil eder. Üçüncü parametre ise butona başlık verir *(bkz. Görsel 74).*



Görsel 74: TclButton Parametreleri

Kod 86'da bir form oluşturuldu ve üzerine bir buton eklendi.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum11/Ornek1)

Ana Kod

```
var
  Form1:TCLForm;
  Buton1:TclButton;
{
  Form1 = TCLForm.Create(Self);
  Buton1 = Form1.AddNewButton(Form1,'Buton1', 'Başlat');
  Form1.Run;
}
```

Kod 86: TclButton Örneği

Kod çalıştırıldığında *Görsel 75*'teki gibi bir çıktı alınır.



Görsel 75: TclButton Kullanımı

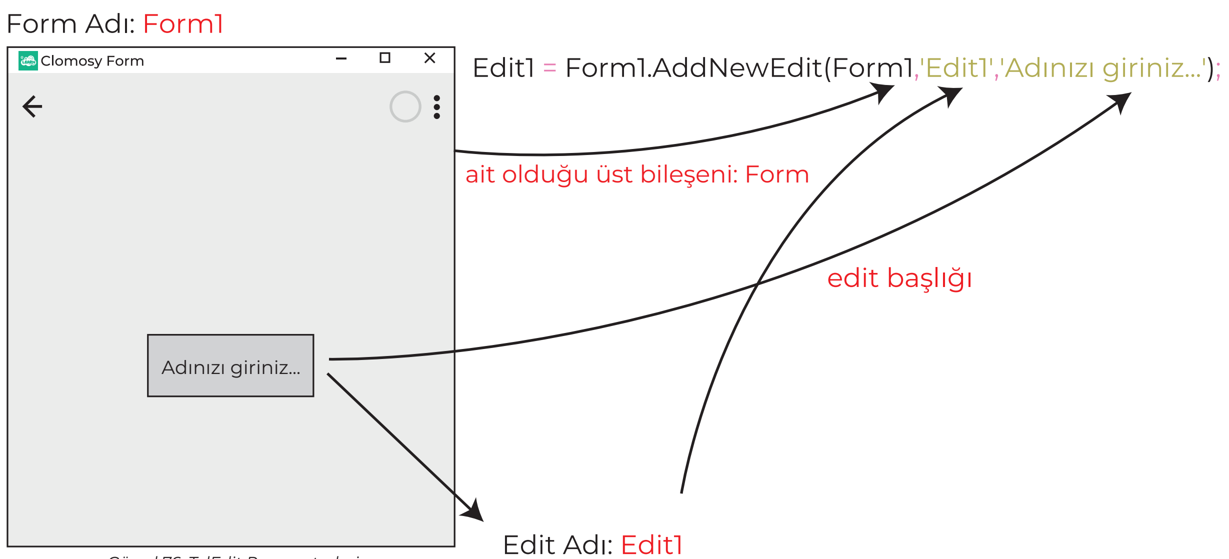
Görüldüğü gibi nesneye hiçbir özellik verilmeden formun merkezinde oluşturuldu. Aynı zamanda yükseklik ve genişliği de otomatik atandı.

Demek ki, bir bileşen standart olarak oluşturulduğunda, hizalama, genişlik ve yükseklik değerleri otomatik olarak atanır. Birçok bileşende de aynı özellik geçerlidir. Bu özellikler sonrasında değiştirilebilir. Bu konular, bileşen özelliklerinde ele alınacaktır.

TCLEDIT

Kullanıcıların form üzerinde metin girmesini sağlamak için kullanılan metin kutusu *TclEdit* bileşenidir. TclEdit bileşeni, form üzerine yerleştirildiğinde kullanıcıların bilgi girmesi istenilen alanı oluşturur. Örneğin; kullanıcı adı veya şifre girişi alanı olarak kullanılabilir.

TclEdit nesnesi oluşturmak için *AddNewEdit* metodu kullanılır. Bu işlev, üç parametre alır. İlk parametre, tanımlanacak olan edit bileşeninin ait olduğu üst bileşeni (parent) belirler. İkinci parametre, editin adını, üçüncü parametre ise nesnenin bilgi metnini ifade eder.



Görsel 76: TclEdit Parametreleri

QR Okut

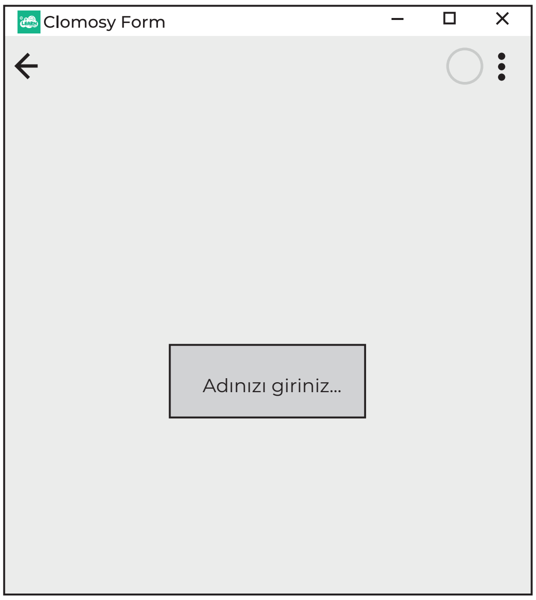
(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum11/Ornek2)

Ana Kod

```
var
  Form1 : TclForm;
  Edit1: TclEdit;
{
  Form1 = TclForm.Create(Self);
  Edit1 = Form1.AddNewEdit(Form1, 'Edit1','Adınızı giriniz...');
  Form1.Run;
}
```

Kod 87: TclEdit Örneği

Kod 87'de isim bilgisini almak isteyen bir edit nesnesi oluşturuldu. *Görsel 77*'de ise çalıştırdıktan sonraki hâli gösterildi.



Görsel 77: TclEdit Kullanımı

TCLLABEL

Kullanıcı arayüzünde açıklama veya bilgi vermek amacıyla kullanılan bir metin etiketi oluşturur. *TclLabel,* form üzerindeki diğer bileşenleri etiketlemek veya kullanıcıya bilgi vermek için tercih edilir. Örneğin; bir giriş alanının yanında "Kullanıcı Adı" veya "Şifre" gibi açıklamalar gösterilebilir.

TclLabel, *AddNewLabel* metodu ile oluşturulur. Bu işlev üç parametre alır; ilk parametre, tanımlanacak olan etiketin ait olduğu üst bileşeni (parent), ikinci parametre, tanımlanan etiketin adını, üçüncü parametre ise etikete atanacak

Kod 88'de, kullanıcının hangi bilgiyi gireyeceğini gösterebilmek için "KULLANICI BİLGİLERİ" başlığı oluşturuldu *(bkz. Görsel 78)*.

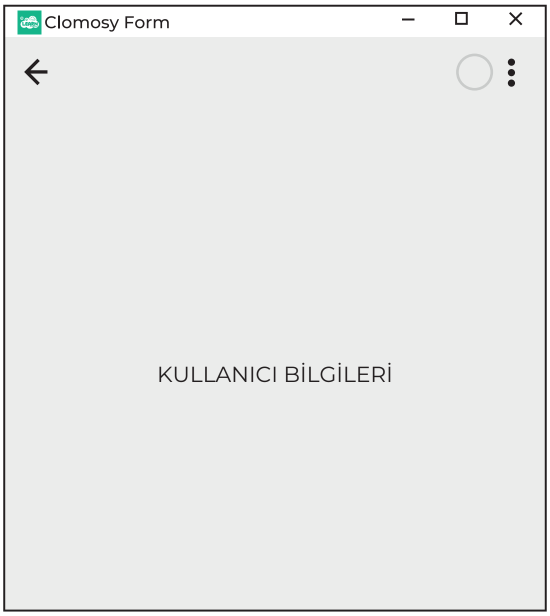
QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum11/Ornek3)

Ana Kod

```
var
  Form1 : TclForm;
  Label1 : TclLabel;
{
  Form1 = TclForm.Create(Self);
  Label1 = Form1.AddNewLabel(Form1,'Label1','KULLANICI BİLGİLERİ');
  Label1.Width = 200;
  Form1.Run;
}
```

Kod 88: TclLabel Örneği



Görsel 78: TclLabel Kullanımı

TCLIMAGE

TclImage, bir formun üzerine resim eklemeye yarar. Uygulamaya görsel eklenmesi, uygulamanın daha bilgilendirici, estetik ve kullanıcı dostu olmasını sağlar. Örneğin, bir uygulamanın logosunu veya açıklama grafiğini göstermek için kullanılabilir.

Görsel nesnesi oluşturmak için *AddNewImage* metodu kullanılır. Bu işlev iki parametre alır. İlk parametre, tanımlanacak olan resim nesnesinin ait olduğu üst bileşenini (parent), ikinci parametre tanımlanan resim nesnesinin adını temsil eder.

Görsel nesnesi oluşturmak için *AddNewImage* metodu kullanılır. Bu metod iki parametre alır. İlk parametre, tanımlanacak olan resim nesnesinin ait olduğu üst bileşenini (parent), ikinci parametre tanımlanan resim nesnesinin adını temsil eder.

TclImage nesnesine resim yüklemek için formun setImage metodu kullanılır *(bkz. Kod 89).*

QR Okut

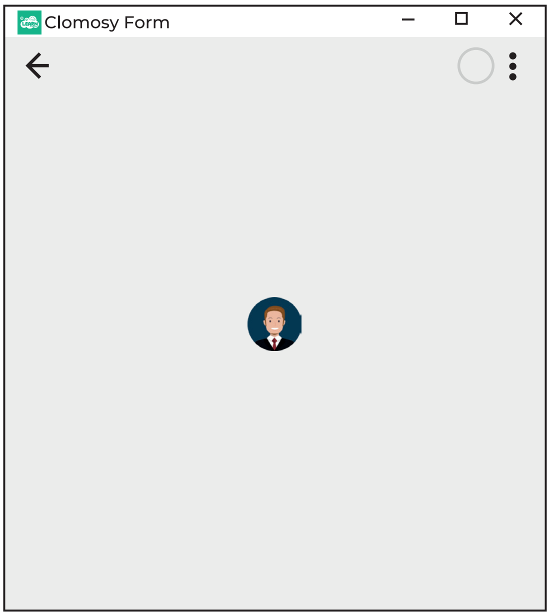
(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum11/Ornek4)

Ana Kod

```
var
 Form1 : TclForm;
 Img1  : TclImage;
{
 Form1 = TclForm.Create(Self);
 Form1.AddAssetFromURL('https://clomosy.com/learn/avatar.png');
 Img1 = Form1.AddNewImage(Form1, 'Img1');
 Form1.setImage(Img1,'avatar.png');
 Form1.Run;
}
```

Kod 89: TclImage Örneği

Görsel 79'da kodun çıktısı yer almaktadır.



Görsel 79: TclImage Kullanımı

TCLPANEL

Form üzerinde belirli bir alanı gruplayarak diğer bileşenleri düzenlemek ve yönetmek için kullanılır. *TclPanel*, kullanıcı arayüzünde belirli bir bölümü ayırmak ve bu bölüme diğer bileşenleri (butonlar, metin kutuları, etiketler vb.) yerleştirmek amacıyla tercih edilir. Böylece, form üzerindeki bileşenlerin düzeni ve görünümü daha kolay kontrol edilir.

TclPanel nesnesi *AddNewPanel* metodu ile oluşturulur. İki parametre alan bu işlevde ilk parametre; panelin ait olduğu üst bileşenini (parent) belirler. İkinci parametre, tanımlanan panelin adını verir *(bkz. Kod 90).*

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum11/Ornek5)

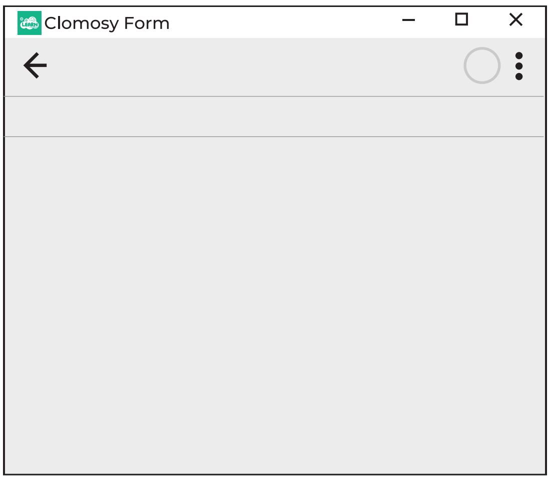
Ana Kod

```
Var
 Form1:TclForm;
 Panel1 : TclPanel;
{
 Form1 = TclForm.Create(self);
 Panel1 = Form1.AddNewPanel(Form1,'Panel1');
 Form1.Run;
}
```

Kod 90: TclPanel Örneği

BİLGİLENDİRME:
Oluşturulan panel nesnesinin kenarlıkları sadece Windows üzerinde görünür. Diğer platformlarda (Android, iOS) kenarlıklar görünmez.

Görsel 80'de Windows üzerinde çalıştırılan kod çıktısı gösterildi.



Görsel 80: TclPanel Kullanımı

Kod 91'de, panel üzerine bir tane buton eklendi. Bunun için bir panel oluşturuldu ve butonun ilk parametresine panel nesnesinin adı yazıldı. Bu sayede butonun ait olduğu üst bileşeni panel olarak ayarlandı ve buton, panel içerisine konumlandırıldı.

QR Okut

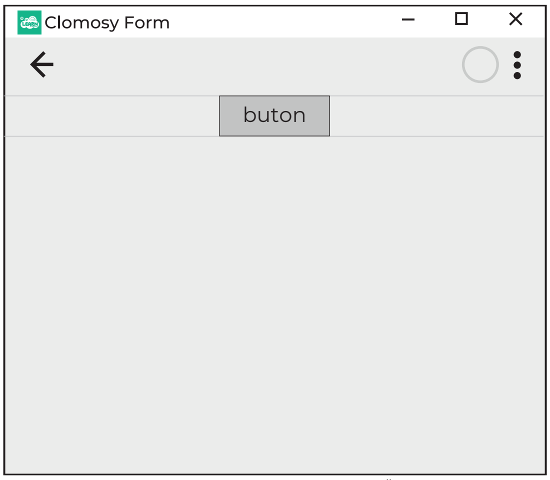
(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum11/Ornek6)

Ana Kod

```
Var
  Form1:TclForm;
  Panel1 : TclPanel;
  Buton1 : TclButton;
{
  Form1 = TclForm.Create(self);
  Panel1 = Form1.AddNewPanel(Form1,'Panel1');
  Buton1 = Form1.AddNewButton(Panel1,'Buton1','buton');
  Form1.Run;
}
```

Kod 91: TclPanel ve TclButton Kullanımı

Kod 91 çalıştırıldığında *Görsel 81*'deki görüntü ekrana gelir.



Görsel 81: TclPanel ve TclButton Örneği

TCLLAYOUT

Form üzerindeki diğer bileşenleri panelde olduğu gibi düzenlemeye yarar. Formda yer alan butonlar, etiketler ve diğer görsel elemanları belirli bir düzene sokmak için kullanılır. Bu düzen, bileşenlerin boyutlarını ve konumlarını otomatik olarak ayarlayarak formun daha düzenli ve estetik görünmesini sağlar. *TclLayout,* genellikle dinamik kullanıcı arayüzleri oluştururken ve farklı ekran boyutlarına uyum sağlarken çok kullanışlıdır.

AddNewLayout metodu ile nesne oluşturulur. Metot, iki parametre alır. İlk parametre; tanımlanacak Layout'un ait olduğu üst bileşenini, ikinci parametre ise tanımlanan layout adını temsil eder.

Kod 92'de bir layout oluşturuldu ve içerisine bir edit nesnesi eklendi.

QR Okut

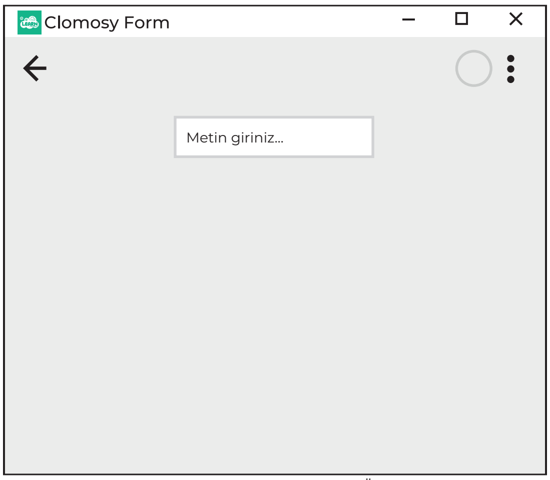
(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum11/Ornek7)

Ana Kod

```
Var
  Form1 : TclForm;
  Layout1 : TclLayout;
  Edit1 : TclEdit;
{
  Form1=TclForm.Create(self);
  Layout1 = Form1.AddNewLayout(Form1,'Layout1');
  Edit1 = Form1.AddNewEdit(Layout1,'Edit1','Metin giriniz...');
  Edit1.Width = 150;
  Form1.Run;
}
```

Kod 92: TclLayout ve TclEdit Kullanımı

Görsel 82'de kodun çıktısı yer alıyor.



Görsel 82: TclLayout ve TclEdit Örneği

TCLHORZSCROLLBOX

Yatay olarak kaydırılabilir bir alan sağlar. İçine birçok bileşen eklenebilir ve bu bileşenlerin bir kısmı ekranda görünmeyebilir. *TclHorzScrollBox*, yatay kaydırma çubuğu sayesinde, ekranda görünmeyen bileşenlere ulaşmak için kaydırma işlemi yapılmasına olanak tanır. Bu, uzun içerikleri veya geniş düzenleri tek bir ekranda gösterebilmek için kullanışlıdır.

Kod 93'te bir TclHorzScrollBox nesnesi oluşturuldu ve içine 20 adet görsel eklendi.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum11/Ornek8)

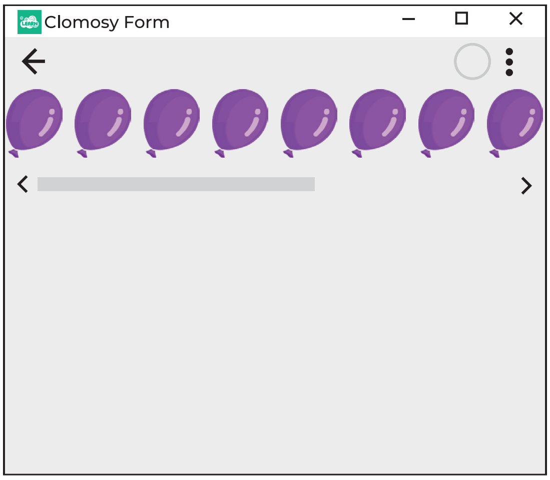
Ana Kod

```
var
 Form1:TclForm;
 HorzScrollBox : TClHorzScrollBox;
 Img1 : TClImage;
 i : Integer;
{
 Form1 = TclForm.Create(Self);
 Form1.AddAssetFromUrl('https://clomosy.com/demos/balloon.png');
 HorzScrollBox = Form1.AddNewHorzScrollBox(Form1,'HorzScrollBox');
 HorzScrollBox.Align = alMostTop;
 HorzScrollBox.Height = 100;

 for (i = 1 to 20)
 {
  Img1 = Form1.AddNewImage(HorzScrollBox,'Img1'+IntToStr(i));
  Img1.Align = alLeft;
  Form1.SetImage(Img1,'balloon.png');
 }
 Form1.Run;
}
```

Kod 93: TclHorzScrollBox ve TclImage Kullanımı

Görsel 83'te kodun çıktısı yer alıyor.



Görsel 83: TclHorzScrollBox ve TclImage Örneği

TCLVERTSCROLLBOX

Dikey olarak kaydırılabilir bir alan sağlar. İçine birçok bileşen eklenebilir ve bu bileşenlerin bir kısmı ekranda görünmeyebilir. *TclVertScrollBox,* dikey kaydırma çubuğu sayesinde, ekranda görünmeyen bileşenlere ulaşmak için kaydırma işleminin yapılmasına olanak tanır. Bu, uzun listeler veya yüksek içerikler gibi dikeyde fazla yer kaplayan düzenleri tek bir ekranda gösterebilmek için kullanışlıdır.

Kod 93, dikey olarak kaydırılabilir bir hâle getirildi *(bkz. Görsel 84).*

QR Okut

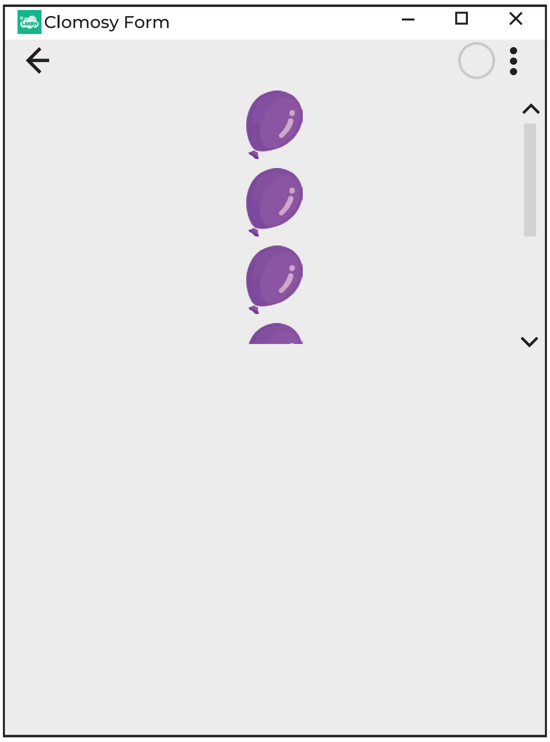
(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum11/Ornek9)

Ana Kod

```
var
 Form1:TclForm;
 VertScrollBox : TClVertScrollBox;
 Img1 : TClImage;
 i : Integer;
{
 Form1 = TclForm.Create(Self);
 Form1.AddAssetFromUrl('https://clomosy.com/demos/balloon.png');
 VertScrollBox = Form1.AddNewVertScrollBox(Form1,'VertScrollBox');
 VertScrollBox.Align = alMostTop;
 VertScrollBox.Height = 100;

 for (i = 1 to 20)
 {
  Img1 = Form1.AddNewImage(VertScrollBox,'Img1'+IntToStr(i));
  Img1.Align = alTop;
  Form1.SetImage(Img1,'balloon.png');
 }
 Form1.Run;
}
```

Kod 94: TclVertScrollBox – TclImage Kullanımı



Görsel 84: TclVertScrollBox ve TclImage Örneği

BİLEŞENLERİN ORTAK ÖZELLİKLERİ

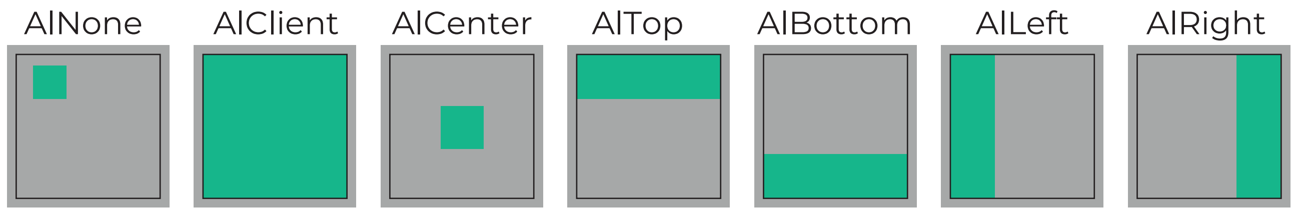
Bileşenlerin nasıl tanımlandığı ve kullanıldığı ele alındı. Nesnelerin konumları, görünümü ve daha birçok özelliği vardır.

Bileşenlerin bazıları arasında ortak özellikler bulunmaktadır. En yaygın ortak özellikler arasında; *Text* (bileşende görüntülenen metin), *Enabled* (bileşenin etkin olup olmadığını belirler), *Visible* (bileşenin görünürlüğünü kontrol eder), *Height* ve *Width* (bileşenin boyutlarını belirler) bulunur. Bu ortak özellikler, bileşenlerin görünümünü ve davranışını özelleştirmeyi kolaylaştırır. Böylece geliştiriciler kullanıcı dostu ve işlevsel arayüzler oluşturabilir.

ALIGN

Bir bileşenin, ait olduğu üst bileşende (örneğin, bir form veya panelde) nasıl yerleştirileceğini belirler. Bu özellik sayesinde bileşenler, kapsayıcının sol, sağ, üst veya alt kenarına sabitlenebilir ya da tüm alanı kaplayacak şekilde ayarlanabilir.

Görsel 85'te en yaygın kullanılan özellikler yer almaktadır.



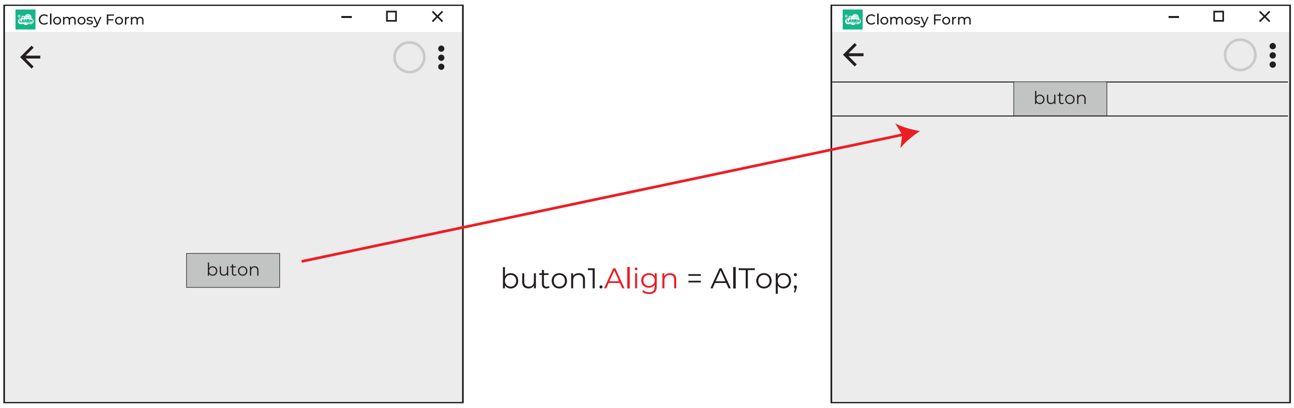
Görsel 85: Align Özellikleri

Örneğin; bir butonun *Align* özelliği *alTop* olarak ayarlandığında buton formun üst kısmına yerleşir ve formun genişliği boyunca uzanır. Bu özellik, kullanıcı arayüzünü dinamik ve esnek bir şekilde düzenlemeyi sağlar. Özellikle farklı ekran boyutlarına ve düzenlerine uyum sağlamak için kullanışlıdır.

Bu özelliği kullanabilmek için hangi nesne kullanılacaksa aşağıdaki gibi tanımlanması gerekmektedir. Aşağıdaki örnek kod bir butonda Align tanımlamasını ifade eder.

```
Button1.Align = AlTop;
```

Bir nesne oluşturulduğu anda çoğu bileşenlerde varsayılan olarak *AlCenter'* dir. Align özelliği AlTop olarak ayarlandığında görüntü aşağıdaki gibi olacaktır.



Görsel 86: TclButton - Align Özelliği Kullanımı

Görsel 86'daki gibi kullanılmak istendiğinde *Kod 95* çalıştırılır.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum11/Ornek10)

Ana Kod

```
Var
  Form1:TclForm;
  Buton1 : TclButton;
{
  Form1 = TclForm.Create(self);
  Buton1 = Form1.AddNewButton(Form1,'Buton1','buton');
  Buton1.Align = alTop;
  Form1.Run;
}
```

Kod 95: TclButton – Align Özellik Örneği

HEIGHT

Bileşenin yüksekliğini piksel cinsinden belirler. Bu özellik, bileşenin dikey boyutunun ayarlanmasına olanak tanır.

Örneğin, bir butonun *Height* değeri 100 olarak ayarlandığında, buton 100 piksel yüksekliğinde olur. Height özelliği, kullanıcı arayüzünde bileşenlerin boyutlarını özelleştirerek, tasarımın estetik ve kullanışlı olmasını sağlar. Bu özellik, form üzerindeki tüm bileşenler için geçerlidir. Bileşenin daha rahat kullanılmasını ve görünmesini sağlamak için ayarlanabilir.

DİKKAT!

Align özelliğinde **AlRight** ve **AlLeft** olarak ayarlandığında Height özelliği kullanılamamaktadır.

Örnek olarak buton nesnesine yükseklik verildi *(bkz. Kod 96).*

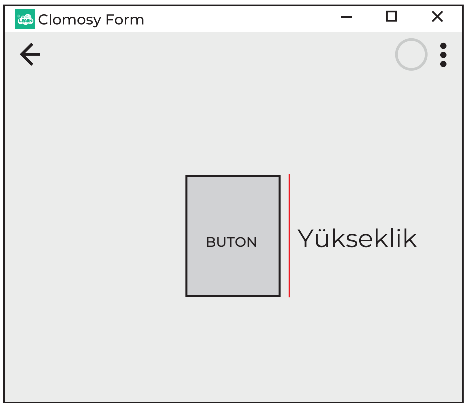
QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum11/Ornek11)

Ana Kod

```
Var
 Form1:TclForm;
 Buton1 : TclButton;
{
 Form1 = TclForm.Create(self);
 Buton1 = Form1.AddNewButton(Form1,'Buton1','BUTON');
 Buton1.Height = 100;
 Form1.Run;
}
```

Kod 96: TclButton – Height Özellik Örneği



Görsel 87: TclButton - Height Özelliği Kullanımı

WIDTH

Bileşenin genişliğini piksel cinsinden belirler. Bu özellik, bileşenin yatay boyutunun ayarlanmasını sağlar.

Örneğin, bir butonun *Width* değeri 200 olarak ayarlandığında buton 200 piksel genişliğinde olur. Width özelliği, kullanıcı arayüzünde bileşenlerin boyutlarını özelleştirerek tasarımın estetik ve kullanışlı olmasını sağlar. Bu özellik, form üzerindeki tüm bileşenler için geçerlidir. Bileşenin daha rahat kullanılmasını ve görünmesini sağlamak içindir.

DİKKAT!
Align özelliğinde **AlTop** ve **AlBottom** olarak ayarlandığında Width özelliği kullanılamamaktadır.

Örnek olarak buton nesnesine genişlik verildi *(bkz. Kod 97)*.

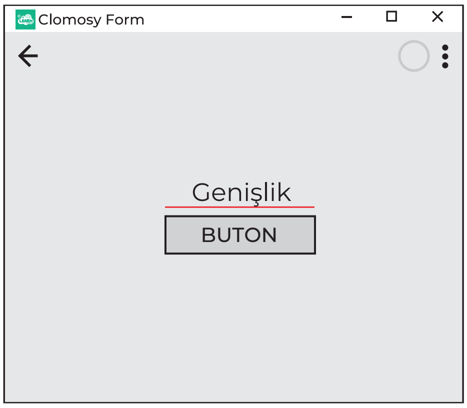
QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum11/Ornek12)

Ana Kod

```
Var
  Form1:TclForm;
  Buton1 : TclButton;
{
  Form1 = TclForm.Create(self);
  Buton1 = Form1.AddNewButton(Form1,'Buton1','BUTON');
  Buton1.Width = 200;
  Form1.Run;
}
```

Kod 97: TclButton – Width Özellik Örneği

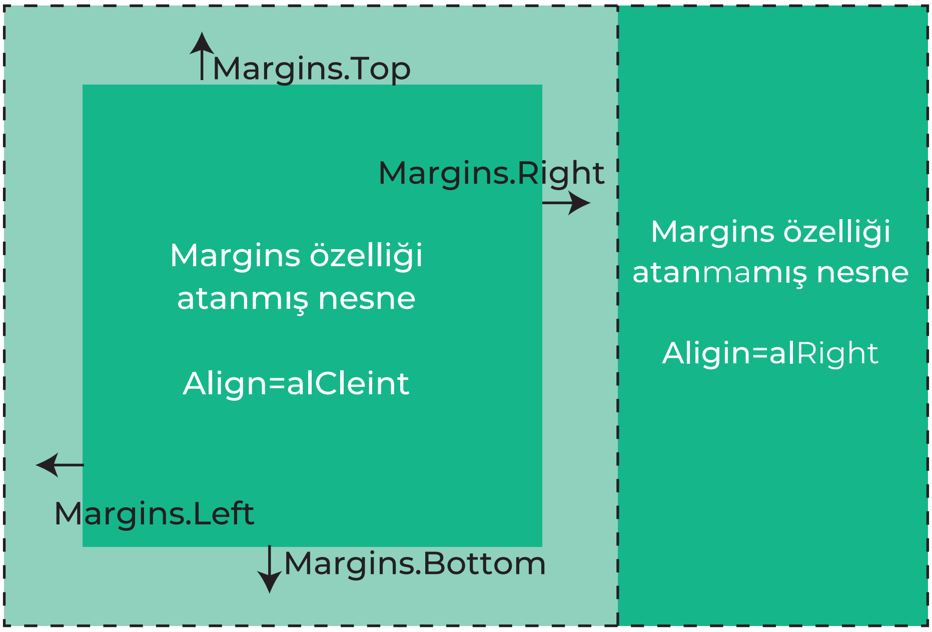


Görsel 88: TclButton – Width Özelliği Kullanımı

MARGINS

Bileşenin etrafındaki boşlukları belirlemek için kullanılır. Bu özellik, bileşenin kapsayıcısının (ait olduğu üst bileşenin) kenarlarından ne kadar uzaklıkta olacağını belirler.

Margins özelliği, dört kenar için ayrı ayrı değerler belirlenmesine olanak tanır: Sol (Left), üst (Top), sağ (Right) ve alt (Bottom) *(bkz. Görsel 89)*. Örneğin; bir butonun *Margins.Left* ve *Margins.Top* değerleri 10 piksel olarak ayarlandığında buton kapsayıcının sol ve üst kenarlarından 10 piksel uzakta yerleştirilir. Bu özellik, bileşenler arasında düzenli ve estetik boşluklar oluşturarak kullanıcı arayüzünün daha düzenli ve okunabilir olmasını sağlar.



Görsel 89: Margins Özelliği

Örneğin bir form içerisine buton tanımlandı ve *Align* özelliği *AlClient* olarak ayarlandı. *Margins* özelliği ile dört yanından 30 piksellik bir boşluk bırakıldı *(bkz. Kod 98).*

QR Okut

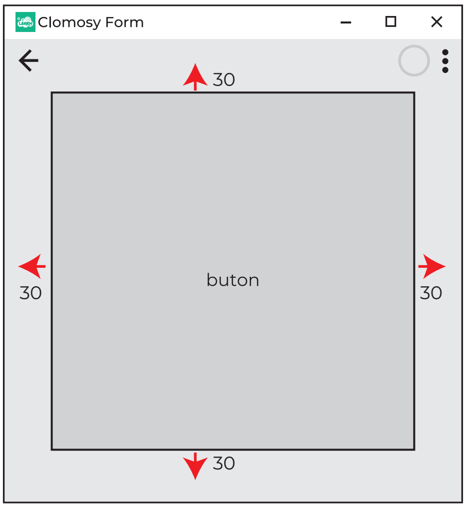
(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum11/Ornek13)

Ana Kod

```
Var
  Form1:TclForm;
  Buton1 : TclButton;
{
  Form1 = TclForm.Create(self);
  Buton1 = Form1.AddNewButton(Form1,'Buton1','BUTON');
  Buton1.Align = alClient;
  Buton1.Margins.Left = 30;
  Buton1.Margins.Right = 30;
  Buton1.Margins.Top = 30;
  Buton1.Margins.Bottom = 30;
  Form1.Run;
}
```

Kod 98: TclButton – Margins Özellik Örneği

Görsel 90'daki gibi bir görüntü elde edilir.



Görsel 90: TclButton – Margins Özelliği Kullanımı

PADDING

Bileşenin iç kenarları ile içindeki içerik arasında bırakılacak boşlukları belirler. Bu özellik, bileşenin içindeki içeriklerin (örneğin, bir panelin içindeki diğer bileşenler) kenarlardan ne kadar uzakta olacağını belirler: Sol (Left), üst (Top), sağ (Right) ve alt (Bottom). Örneğin; bir panelin *Padding.Left* ve *Padding.Top* değerleri 10 piksel olarak ayarlanırsa panelin içindeki içerik sol ve üst kenarlardan 10 piksel uzakta yerleştirilir. Bu özellik, bileşenin içindeki içeriklerin sıkışık görünmesini önleyerek kullanıcı arayüzünün daha rahat okunabilir ve estetik olmasını sağlar.

Margins özelliğinde içteki nesne mesafeyi belirlerken, *padding* özelliğinde ise dıştaki nesne kenarlara olan uzaklığı belirlemektedir.

UYARI!
Padding özelliği sadece bileşenlerde kullanılır. Form üzerinde kullanılan bir özellik değildir.

Margins örneği değiştirilerek aynı yapı, padding özelliğine uyarlandı.

Padding örneğine uyarlamak için form içerisine bir panel ve panel içerisine bir buton eklendi. Padding özelliği panel nesnesine verildi *(bkz. Kod 99)*.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum11/Ornek14)

Ana Kod

```
Var
  Form1:TclForm;
  Panel1 : TcLPanel;
  Buton1 : TclButton;
{
  Form1 = TclForm.Create(self);

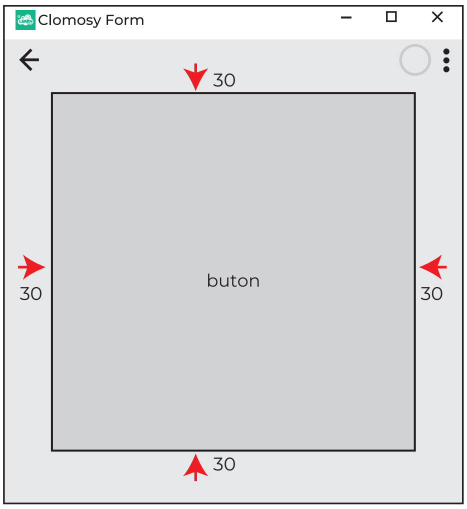
  Panel1 = Form1.AddNewPanel(Form1,'Panel1');
  Panel1.Align = alClient;
  Panel1.Padding.Left = 30;
  Panel1.Padding.Right = 30;
  Panel1.Padding.Top = 30;
  Panel1.Padding.Bottom = 30;

  Buton1 = Form1.AddNewButton(panel1,'Buton1','BUTON');
  Buton1.Align = alClient;

  Form1.Run;
}
```

Kod 99: TclButton – Padding Özellik Örneği

Yapılan örneğin ekran görüntüsü *Görsel 91*'de verildi. Burada panel nesnesinden 30 piksel içteki butona mesafe ayarlandı.



Görsel 91: TclButton – Padding Özelliği Kullanımı

POSITION

Bileşenin form üzerindeki yatay ve dikey konumunu belirler. *Position.X,* bileşenin sol kenarının formun ya da ait olduğu üst bileşenin sol kenarına olan uzaklığını piksel cinsinden, *Position.Y,* bileşenin üst kenarının formun ya da ait olduğu üst bileşenin üst kenarına olan uzaklığını piksel cinsinden belirler.

Örneğin; bir butonun Position.X değeri 50 ve Position.Y değeri 100 olarak ayarlanırsa buton formun sol kenarından 50 piksel ve üst kenarından 100 piksel uzaklıkta yerleştirilir. Bu özellikler, bileşenlerin form üzerindeki kesin konumlarını ayarlayarak kullanıcı arayüzünün düzenini ve görünümünü hassas bir şekilde kontrol etmeyi sağlar.

Kullanımı aşağıdaki gibidir:

```
Buton1.Position.X = 50;
Buton1.Position.Y = 100;
```

BİLGİLENDİRME:
Clomosy'de bir bileşenin konumu, Position.X ve Position.Y özellikleriyle ayarlanmak istendiğinde, Align özelliğinin alNone olarak ayarlanması gereklidir. Eğer Align özelliği alNone olarak ayarlanmazsa bileşenin konumu otomatik olarak ayarlanır ve Position.X ve Position.Y değerleri etkisiz olur.

Örnekte x konumu 50, y konumu 100 olarak ayarlandı *(bkz. Kod 100).*

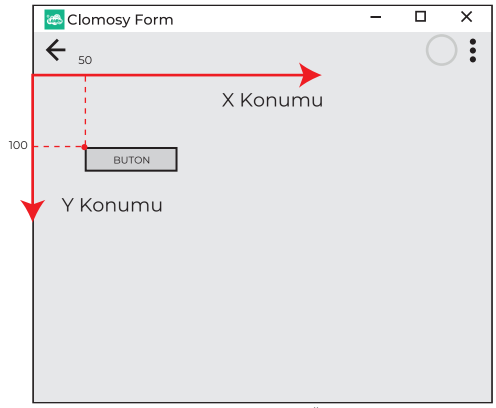
QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum11/Ornek15)

Ana Kod

```
Var
 Form1:TclForm;
 Buton1 : TclButton;
{
 Form1 = TclForm.Create(self);
 Buton1 = Form1.AddNewButton(Form1,'Buton1','BUTON');
 Buton1.Align = alNone;
 Buton1.Position.X = 50;
 Buton1.Position.Y = 100;
 Form1.Run;
}
```

Kod 100: TclButton – Position Özellik Örneği

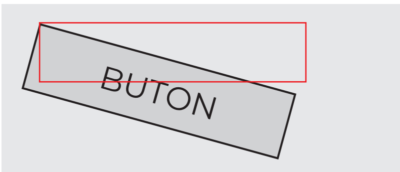


Görsel 92: TclButton – Position Özelliği Kullanımı

ROTATIONANGLE

Bileşenin saat yönünde ne kadar döndürüleceğini derece cinsinden belirler. Bu özellik sayesinde bir bileşen belirli bir açıyla döndürülebilir. Örneğin, bir butonun RotationAngle değeri 20 derece olarak ayarlandığında buton 20 derece saat yönünde döner *(bkz. Görsel 93)*. Bu özellik, özellikle görsel efektler vermek, kullanıcı arayüzünü daha dinamik ve kullanışlı hale getirmek için kullanılır. RotationAngle değeri pozitif olduğunda saat yönünde, negatif olduğunda ise saat yönünün tersine döner. Bu özellik, bileşenin normal konumuna kıyasla nasıl görüneceğinin ve yerleştirileceğinin kontrol edilmesini sağlar.

```
Button1.RotationAngle = 20;
```



Görsel 93: TclButton – RotationAngle Özelliği Kullanımı

QR Okut

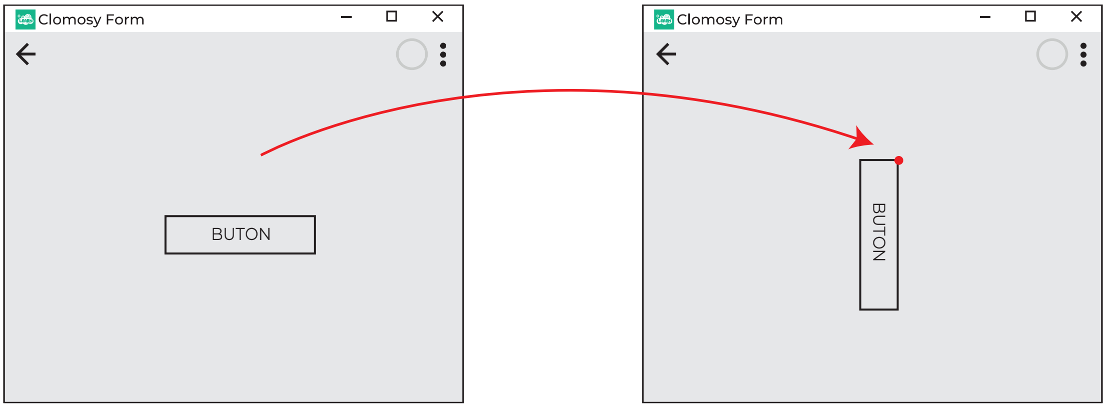
(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum11/Ornek16)

Ana Kod

```
Var
  Form1:TclForm;
  Buton1 : TclButton;
{
  Form1 = TclForm.Create(self);
  Buton1 = Form1.AddNewButton(Form1,'Buton1','BUTON');
  Buton1.RotationAngle = 90;
  Form1.Run;
}
```

Kod 101: TclButton – RotationAngle Özellik Örneği

Görsel 94'te ilk hâli ve özellik kullanıldıktan sonraki hâli gösterilmiştir.



Görsel 94: TclButton – RotationAngle Kullanımı

VISIBLE

Bileşenin ekranda görünüp görünmeyeceğini kontrol eder. Bu özellik True olarak ayarlandığında bileşen ekranda görünür; False olarak ayarlandığında ise bileşen gizlenir. Varsayılan olarak True'dur.

Örneğin, bir butonun Visible değeri False olarak ayarlanırsa, buton kullanıcı tarafından görülemez ve etkileşimde bulunulamaz. Bu özellik, kullanıcı arayüzünde belirli durumlarda bazı bileşenleri gizlemek veya göstermek için kullanılır.

```
Buton1.Visible = False;
```

LOCKED

Bileşenin tasarım zamanında taşınmasını veya boyutlandırılmasını engelleyen bir özelliktir. Varsayılan değeri False'dır. Bu özellik True olarak ayarlandığında bileşen form üzerinde kilitlenir ve yanlışlıkla yerinin değiştirilmesi veya boyutunun ayarlanması önlenir. Özellikle karmaşık kullanıcı arayüzleri oluştururken bileşenlerin yanlışlıkla hareket ettirilmesini engellemek için kullanışlıdır.

Aşağıdaki gibi kullanılmaktadır.

```
Button1.Locked =True;
```

SCALE

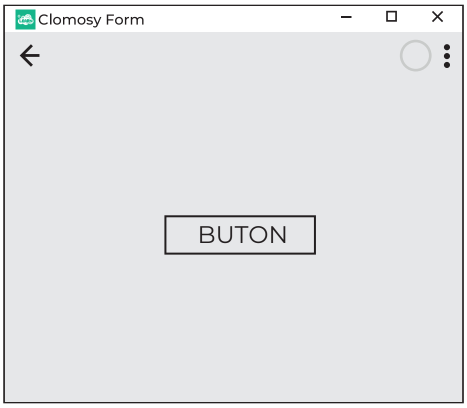
Bileşenin yatay ve dikey boyutlarını orantılı olarak ölçeklendirmek için kullanılır. *Scale.X* özelliği, bileşenin genişliğini; *Scale.Y* özelliği ise bileşenin yüksekliğini belirli bir ölçek faktörü ile çarpar. X ve Y için varsayılan değer 1'dir. Değerlerin 2 olması durumunda bileşen boyutu iki kat artar.

Örneğin, bir bileşenin Scale.X değeri 1.5 olarak ayarlandığında bileşenin genişliği %50 oranında artar. Benzer şekilde, Scale.Y değeri 0.75 olarak ayarlandığında bileşenin yüksekliği %25 oranında azalır.

Bu özellikler, bileşenlerin boyutlarını yatay ve dikey olarak bağımsız şekilde ölçeklendirerek kullanıcı arayüzünü farklı ekran boyutlarına ve çözünürlüklere uyarlamayı kolaylaştırır. Bu sayede bileşenlerin orantılı ve estetik görünmesini sağlar.

```
Buton1.Scale.X = 2;
Buton1.Scale.Y = 1.5;
```

(bkz. Görsel 95)



Görsel 95: TclButton – Scale Özelliği Kullanımı

Kod 102'de uygulama içerisinde, buton nesnesi (Buton1) formun en altına ve boyutu iki kata çıkarıldı. Görsel nesnesi (Image1) ise formun merkezinde olacak şekilde yükseklik ve genişliği ayarlandı. Son olarak metin girişi (Edit1) için oluşturulan nesne, formun en üstüne sabitlendi, sağ ve sol kenarlarından boşluk bırakıldı.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum11/Ornek17)

Ana Kod

```
var
 Form1 : TclForm;
 Buton1 : TclButton;
 Image1 : TclImage;
 Edit1 : TclEdit;
{
 Form1 = TclForm.Create(Self);

 Buton1 = Form1.AddNewButton(Form1,'Buton1','Buton');
 Buton1.Align = AlBottom;
 Buton1.Scale.X = 2;
 Buton1.Scale.Y = 2;

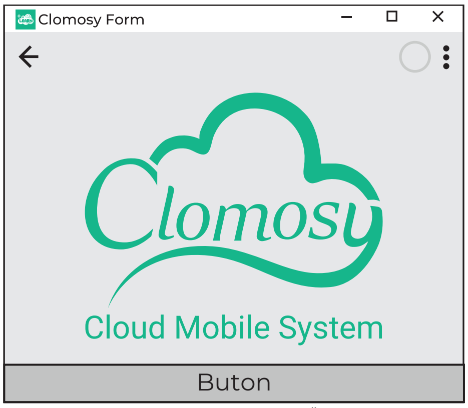
 Image1 = Form1.AddNewImage(Form1,'Image1');
 Form1.setImage(Image1,' https://clomosy.com/demos/clomosyinfo.png');
 Image1.Align = alClient;
 //Image1.Visible = True //False yaparsak resim görünmez
 Image1.Height = 400;
 Image1.Width = 500;

 Edit1 = Form1.AddNewEdit(Form1,'Edit1','Yaz');
 Edit1.Align = AlTop;
 Edit1.Margins.Left = 100;
 Edit1.Margins.Right = 100;
 Edit1.Width = 100;

 Form1.Run;
}
```

Kod 102: TclButton – Scale Özellik Örneği

Kod 102 çalıştırıldığında *Görsel 96*'daki görüntü ile karşılaşılır.



Görsel 96: TclButton – Scale Örneği

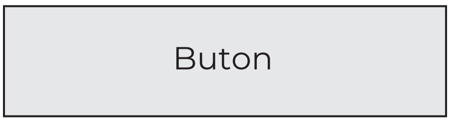
ENABLED

Bileşenin etkin olup olmadığını belirler. Bu özellik True olarak ayarlandığında, bileşen kullanıcı etkileşimlerine açıktır ve kullanılabilir durumdadır; False olarak ayarlandığında ise bileşen devre dışı kalır ve kullanıcı tarafından tıklanamaz, seçilemez veya herhangi bir şekilde etkileşimde bulunulamaz. Varsayılan olarak True'dir.

Örneğin, bir butonun *Enabled* değeri False olarak ayarlandığında, buton gri renkte görünür ve tıklanamaz hâle gelir. Bu özellik, kullanıcı arayüzünde belirli koşullar altında bazı bileşenlerin kullanılmasını engellemek veya etkinleştirmek için kullanılır. Böylece uygulamanın davranışını dinamik olarak kontrol etmeyi sağlar.

```
Buton1.Enabled = False;
```

Görsel 97'deki gibi görünmektedir.



buton1.Enabled = True;



buton1.Enabled = False;

Görsel 97: TclButton – Enabled Özelliği Kullanımı

DİĞER TEMEL BİLEŞEN ÖZELLİKLERİ

TEXT

Bileşenin üzerinde görünen metni belirler. Bu özellik, genellikle TclLabel, TclButon ve TclEdit gibi bileşenlerde kullanılır ve bileşenin kullanıcıya göstereceği yazının ayarlanmasını sağlar. Aynı zamanda bileşen içindeki metin değerini almak için de kullanılır.

Örneğin, bir butonun *Text* özelliği "Gönder" olarak ayarlanırsa buton üzerinde "Gönder" yazısı görünür. Benzer şekilde, bir metin kutusunun Text özelliği "Merhaba" olarak ayarlandığında metin kutusunda "Merhaba" yazısı yer alır. Text özelliği, kullanıcı arayüzündeki metinlerin dinamik olarak değiştirilmesine ve bileşen içeriğinin kullanıcıya uygun şekilde sunulmasına olanak tanır.

Kod 103'ün başlangıcında butonun Text özelliği "Buton" olarak tanımlandı. Ardından Text özelliği ile "Merhaba" olarak değiştirildi ve uygulama açıldığında "Merhaba" metni görüldü.

Ek olarak ShowMessage içerisinde butonun üzerindeki metne erişim sağlandı.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum11/Ornek18)

Ana Kod

```
var
  Form1 : TclForm;
  Buton1 : TclButton;
{
  Form1 = TclForm.Create(Self);
  Buton1 = Form1.AddNewButton(Form1,'Buton1','Buton');
  Buton1.Text = 'Merhaba';
  ShowMessage(Buton1.Text);
  Form1.Run;
}
```

Kod 103: TclButton – Text Özellik Örneği

CAPTION

Bileşenin üzerinde görünen metni belirler. Bu özellik, bileşenin kullanıcıya ne tür bir bilgi sunduğunu veya ne işlevi olduğunu göstermek için metin eklenmesini sağlar.

Örneğin; bir butonun *Caption* özelliği "Kaydet" olarak ayarlandığında, buton üzerinde "Kaydet" yazısı görünür ve kullanıcı bu butona tıkladığında veriyi kaydedeceğini anlar. Caption özelliği, bileşenlerin üzerindeki metinleri dinamik olarak değiştirmeyi ve kullanıcı arayüzünü daha anlamlı hale getirmeyi sağlar.

Kullanımı aşağıdaki gibidir.

```
Buton1.Caption = 'Merhaba';
ShowMessage(Buton1.Caption);
```

BİLGİLENDİRME:
Text genellikle metin giriş bileşenlerinde kullanılırken Caption daha çok başlık veya etiket gibi bileşenlerde kullanılır ve her iki özellik, bileşenlerin üzerindeki metni kontrol etmek için farklı senaryolarda uygulanır.

CLTYPEOFFIELD

Bir metin giriş bileşeninde istenilen her karakter yazılabilir. Ancak karakter sınırlaması getirmek istendiğinde bu özellik kullanılır. Özelliğe *taFloat* ataması yapıldığında yalnızca sayılar ve virgül karakterleri, *taString* ataması yapıldığında ise tüm karakterler (sayılar, semboller, harfler) yazılır.

```
Edit1.clTypeOfField = taFloat;
```

STYLEDSETTINGS

Bileşenin metin görüntüleme özelliklerini ayarlamak için kullanılır. Bu özellik, metnin yazı stilini (style), boyutunu (size) ve rengini (color) belirler

Bu özelliğin birden fazla kullanımı mevcuttur. Öncelikle bu kullanımları aktifleştirmek için *StyledSettings* özelliği ayarlanmalıdır. Örnek kullanımı;

```
Buton1.StyledSettings = ssFamily;
```

ssFamily değeri, bileşenin aile temelli stil ayarlarını kullanmasını ifade eder. Bu, bileşenin ait olduğu üst bileşeni tarafından tanımlanan stil özelliklerini kullanmasını sağlar. StyledSettings özelliği ayarlaması yapıldıktan sonra özellikler verilebilir.

Metnin boyutu *Font.Size* özelliği ile ayarlanır.

```
Buton1.TextSettings.Font.Size = 20;
```

Bileşenin metin rengi *FontColor* özelliği ile ayarlanır.

```
Buton1.TextSettings.FontColor = clAlphaColor.clHexToColor('#8a067c');
```

Metnin stili *Font.Style* özelliği ile ayarlanır. Bu özellik, metnin üç farklı stil özelliğinin ayarlanmasına olanak tanır:

1. *fsBold:* Metni kalın yapar.
2. *fsItalic:* Metni italik (eğik) yapar.
3. *fsUnderline:* Metnin altını çizer.

Aşağıdaki örnekte bileşen metni italik olarak ayarlanmıştır.

```
Buton1.TextSettings.Font.Style = [fsItalic];
```

Örnekteki gibi tek veya birden fazla özellik verilebilir. Aşağıda hem eğik yazı hem de altı çizili olması sağlanmıştır.

```
Buton1.TextSettings.Font.Style = [fsItalic,fsUnderline];
```

Örnek 1

Bir buton metninin yazı stilini kalın ve italik, boyutunu 16 punto, rengini kırmızı yapmak için TextSettings özelliği kullanılabilir. Bu özellik, bileşene ait metnin görünümünü özelleştirerek kullanıcı arayüzünün daha estetik ve okunabilir olmasını sağlar *(bkz. Kod 104)*.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum11/Ornek19)

Ana Kod

```
var
  Form1 : TclForm;
  Buton1 : TclButton;
{
  Form1 = TclForm.Create(Self);
  Buton1 = Form1.AddNewButton(Form1,'Buton1','Buton');
  Buton1.StyledSettings = ssFamily;
  Buton1.TextSettings.FontColor = clAlphaColor.clHexToColor('#ff0000');
  Buton1.TextSettings.Font.Size = 16;
  Buton1.TextSettings.Font.Style = [fsItalic,fsBold];
  Form1.Run;
}
```

Kod 104: TclButton – StyledSettings Özellik Örneği

HINT

Bileşenin üzerine fareyle gelindiğinde görüntülenen kısa açıklamayı veya ipucu metnini belirler. Bu özellik, kullanıcıya bileşenin işlevi veya kullanımı hakkında ek bilgi verir.

Örneğin; bir düğmenin *Hint* özelliği "Bu düğmeye tıklayarak kaydedin" olarak ayarlanırsa kullanıcı fareyi düğmenin üzerine getirdiğinde bu metin küçük bir kutucukta görünür *(bkz. Görsel 98)*. Hint özelliği, kullanıcı deneyimini iyileştirerek uygulamanın daha sezgisel ve kullanıcı dostu olmasını sağlar *(bkz. Kod 105)*.

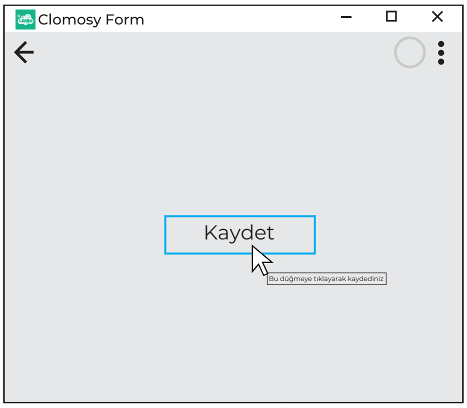
QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum11/Ornek20)

Ana Kod

```
var
  Form1 : TclForm;
  Buton1 : TclButton;
{
  Form1 = TclForm.Create(Self);
  Buton1 = Form1.AddNewButton(Form1,'Buton1','Kaydet');
  Buton1.Hint = 'Bu düğmeye tıklayarak kaydedin';
  Form1.Run;
}
```

Kod 105: TclButton – Hint Özellik Örneği



Görsel 98: TclButton – Hint Özelliği Kullanımı

HITTEST

Bir bileşenin dokunma veya tıklama olayını yakalayıp yakalayamayacağını belirleyen *HitTest* özelliği, bu olaylara tepki verip vermeyeceğini kontrol eder. Bu özellik True olarak ayarlandığında bileşen tıklama ve hareket gibi olayları algılar ve bu olaylara tepki verir. False olarak ayarlandığında ise bileşen, tıklama olaylarını görmezden gelir ve bu olaylara tepki vermez.

Örneğin; bir butonun HitTest özelliği False olarak ayarlanırsa kullanıcı düğmeye tıkladığında hiçbir şey olmaz. Kullanımı aşağıdaki gibidir.

```
Buton1.HitTest = False;
```

OPACITY

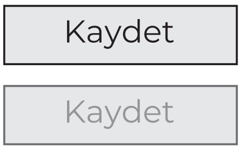
Bileşenin ne kadar saydam olduğunu belirleyen *Opacity* özelliği, bileşenin opaklık seviyesini 0 ile 1 arasında bir değer olarak ayarlar. Burada 0, bileşenin tamamen saydam (görünmez) olduğu; 1, bileşenin tamamen opak (görünür) olduğu anlamına gelir. Varsayılan değeri 1’dir.

Not:
Opacity ve Visible özellikleri arasındaki en önemli fark; çalışma anında bir nesnenin Visible değeri false olduğunda nesneye ait *tbeOnClick* gibi kullanıcı etkileşimli olaylar çalışmaz. Opacity değeri sıfır olan bir nesnede çalışma anında görünmez fakat tbeOnClick özelliği kullanılabilir.

Örneğin, bir bileşenin Opacity değeri 0.5 olarak ayarlanırsa bileşen yarı saydam olur ve arkasındaki öğeler kısmen görülebilir. Opacity özelliği, kullanıcı arayüzünde görsel efektler oluşturmak, belirli bileşenleri vurgulamak veya gizlemek için kullanılır. Böylece daha dinamik ve estetik bir tasarım sağlar.

Kullanımı *Görsel 99*'daki gibidir.

```
Buton1.Opacity = 0.5;
```



Görsel 99: TclButton – Opacity Özelliği Kullanımı

SETFOCUS

Bileşenin klavye odağını almasını sağlar. Bu yöntem çağrıldığında, kullanıcı klavyede bir tuşa bastığında girdinin yönlendirileceği bileşen belirlenir.

Örneğin; bir metin kutusunun *SetFocus* yöntemi çağrıldığında kullanıcı metin kutusuna yazı yazmaya başlayabilir. Bu özellik, kullanıcı arayüzünde belirli bir bileşeni ön plana çıkarmak ve kullanıcıdan veri girişi almak için kullanılır. Böylece kullanıcı deneyimi daha akıcı ve yönlendirilmiş hâle gelir.

Kod 106'da edit oluşturuldu ve setFocus özelliği kullanılarak uygulama açıldığında edit nesnesine odaklandı. Bunu görebilmek için mobil cihazda kod çalıştırılabilir.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum11/Ornek21)

Ana Kod

```
var
  Form1 : TclForm;
  Edit1 : TclEdit;
{
  Form1 = TclForm.Create(Self);

  Edit1 = Form1.AddNewEdit(Form1,'Edit1','Metni giriniz...');
  Edit1.setFocus;
  Form1.Run;
}
```

Kod 106: TcEdit – SetFocus Özellik Örneği

READONLY

Bileşenin yalnızca okunabilir olmasını sağlar. Bu özellik True olarak ayarlandığında bileşen içindeki metin veya veri görüntülenebilir ancak kullanıcı tarafından değiştirilemez. Bu özellik TclEdit, TclMemo gibi veri yazılabilen bileşenler için kullanılır.

Örneğin; bir metin kutusunun (TclEdit) *ReadOnly* değeri True olarak ayarlanırsa kullanıcı metin kutusundaki mevcut metni görebilir ancak düzenleyemez. ReadOnly özelliği, kullanıcıya bilgi gösterip bu bilginin yanlışlıkla değiştirilmesini önlemek için kullanılır.

Kod 107 çalıştırılarak nesneye bir şeyler yazılması denenebilir. Bunun yapılamayacağı fark edilir. Bu özellik False yapıldığında artık yazmaya başlanabilir.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum11/Ornek22)

Ana Kod

```
var
  Form1 : TclForm;
  Edit1 : TclEdit;
{
  Form1 = TclForm.Create(Self);
  Edit1 = Form1.AddNewEdit(Form1,'Edit1','Metni giriniz...');
  Edit1.ReadOnly = True;
  Form1.Run;
}
```

Kod 107: TclEdit – ReadOnly Özellik Örneği

MAXLENGTH

Bir metin giriş bileşenine (TclEdit, TclMemo vb.) girilebilecek maksimum karakter sayısını belirler. Bu özellik, veri girişi sırasında belirli bir karakter sınırını koruyarak veri kontrolünü sağlamak ve kullanıcıların çok uzun metinler girmesini engellemek için kullanılır.

Örneğin; bir metin kutusunun *MaxLength* değeri 5 olarak ayarlandığında kullanıcı bu kutuya en fazla 5 karakter girebilir.

Kod 108'de 5 karakterden fazla veri girilemeyeceği gösterilmiştir.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum11/Ornek23)

Ana Kod

```
var
 Form1 : TclForm;
 Edit1 : TclEdit;
{
 Form1 = TclForm.Create(Self);
 Edit1 = Form1.AddNewEdit(Form1,'Edit1','Metni giriniz...');
 Edit1.MaxLength = 5;
 Form1.Run;
}
```

Kod 108: TclEdit – MaxLength Özellik Örneği

KEYBOARDTYPE

Özellikle mobil uygulamalarda, bir metin giriş bileşeninin (TclEdit vb.) hangi tür sanal klavye ile etkileşime gireceğini belirler. Bu özellik, kullanıcıya metin girişi yaparken uygun klavye türünü otomatik olarak gösterir.

KeyboardType değeri aşağıdaki gibi kullanılmaktadır.

`Edit1.KeyboardType = vktEmailAddress;`

Yukarıdaki gibi kullanıldığında e-posta adresi girişi için uygun klavye açılır. Bunun dışında *Tablo 16'*daki özellikler de kullanılabilir.

Tür	Kullanım Amacı
vktDefault	Varsayılan klavye türü
vktNumsAndPunctuation	Sayılar ve noktalama işaretleri için klavye
vktNumberPad	Yalnızca sayılar için sayısal tuş takımı
vktPhonePad	Telefon numarası girişi için klavye
vktAlphabet	Alfabetik karakterler için klavye
vktURL	URL girişi için klavye
vktNamePhonePad	Sayılar ve alfabetik karakterler için klavye
vktDecimalNumberPad	Ondalık sayılar için sayısal klavye

Tablo 16: KeyboardType Türleri

Kod 109, mobil cihazda test edildiğinde e-posta girilebilecek bir klavye açılır.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum11/Ornek24)

Ana Kod

```
var
 Form1 : TclForm;
 Edit1 : TclEdit;
{
 Form1 = TclForm.Create(Self);

 Edit1 = Form1.AddNewEdit(Form1,'Edit1','Metni giriniz...');
 Edit1.KeyboardType = vktEmailAddress;
 Form1.Run;
}
```

Kod 109: TclEdit – keyboardType Özellik Örneği

WITH KONTROL YAPISI

With ifadesi, bir nesnenin birden fazla özelliğini veya yöntemini kısa ve okunabilir bir şekilde kullanılmasına olanak tanır. With ifadesi, belirli bir nesne üzerinde birden fazla işlem yaparken kodu daha temiz ve anlaşılır hale getirir

Örnekte, with kontrol yapısı kullanılmadan bir butona hizalama (Align), yükseklik (Height), genişlik (Width) verildi *(bkz. Kod 110).*

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum11/Ornek25)

Ana Kod

```
var
 Form1 : TclForm;
 Buton1 : TclButton;
{
 Form1 = TclForm.Create(Self);
 Buton1 = Form1.AddNewButton(Form1,'Buton1','Başlat!');
 Buton1.Align = AlCenter;
 Buton1.Width = 250;
 Buton1.Height = 200;
 Form1.Run;
}
```

Kod 110: TclButton Kullanımı

With kontrol yapısı ile *Kod 110'*daki örnek nasıl en kısa ve okunabilir şekilde tanımlanır gösterildi *(bkz. Kod 111).*

> ***UYARI:***
> *With* yapısı kullanılmadan önce nesnenin oluşturulması gerekmektedir.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum11/Ornek26)

Ana Kod

```
var
 Form1 : TclForm;
 Buton1 : TclButton;
{
 Form1 = TclForm.Create(Self);
 Buton1 = Form1.AddNewButton(Form1,'Buton1','Başlat!');
 With Buton1 do
 {
  Align = AlCenter;
  Width = 150;
  Height = 200;
 }
 Form1.Run;
}
```

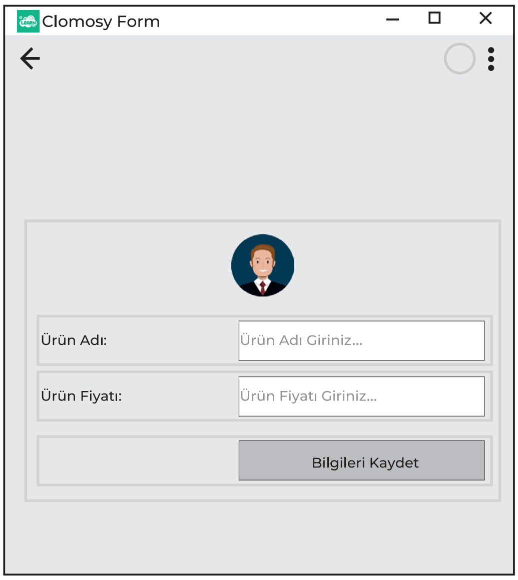
Kod 111: With Kontrol Yapısı Örneği

Alıştırmalar

1. TclButton bileşeni oluşturunuz ve üzerine tıklandığında bir mesaj kutusu gösteriniz.
2. TclEdit bileşeni oluşturunuz ve içine bir metin girildiğinde metni bir TclLabel bileşenine yazdırınız.
3. TclLabel bileşeni oluşturunuz ve başlangıçta "Merhaba Dünya!" yazısını gösteriniz.
4. TclImage bileşeni oluşturunuz ve bir resim dosyasını bu bileşen içerisinde gösteriniz.
5. TclPanel bileşeni oluşturunuz ve içerisine bir TclButton bileşeni ekleyiniz.
6. TclLayout bileşeni oluşturunuz ve içine iki TclButton bileşeni yerleştiriniz.
7. TclButton bileşenine tıklandığında TclEdit bileşenindeki metni TclLabel bileşenine kopyalayınız.
8. TclImage bileşenine tıklandığında mesaj kutusu gösteriniz.
9. TclButton bileşenine tıklandığında TclPanel içerisindeki tüm bileşenleri gizleyiniz.
10. TclButton bileşenine tıklandığında TclEdit bileşeninin boş olup olmadığını kontrol edin ve uygun bir mesaj gösteriniz.
11. TclButton bileşenine tıklandığında TclImage bileşenindeki resmi değiştiriniz.
12. TclButton bileşenine tıklandığında TclLayout içindeki tüm bileşenlerin boyutunu değiştiriniz.
13. TclButton bileşenine tıklandığında formu kapatınız.
14. TclButton bileşenine tıklandığında TclEdit bileşenini temizleyiniz.
15. TclButton bileşenine tıklandığında TclLabel bileşeninin fontunu değiştiriniz.

16. TclButton bileşenine tıklandığında TclLabel bileşenini gizleyiniz ya da gösteriniz.

17. Kullanıcıdan bir ürün adı, fiyat bilgisi ve ürün görseli alarak bunları kaydetmek için kaydet butonu arayüzü tasarlayınız *(bkz. Görsel 100)*. Bu bileşenlerin hepsini bir panel içerisine alınız.



Görsel 100: 17.Soru Arayüz Tasarımı

18. Ürün kataloğu uygulaması tasarlayınız. Bu uygulama, kullanıcıya ürün görselleri ve her bir ürünle ilgili açıklamalar sunmalıdır. Arayüzde 10 adet görsel yer almalı ve bu görseller dikey olarak kaydırılabilir olmalıdır. Her görselin yanında, ürünle ilgili bilgilendirme (isim, fiyat, açıklama gibi) içeren bir kutucuk yer almalıdır. Tasarımda kaydırma çubukları kullanarak eklenen ürünlerin kolayca görüntülenmesi sağlanmalıdır *(bkz. Görsel 101)*.



Görsel 101: 18.Soru Arayüz Tasarımı

19. 5 adet butonun alt alta opaktan saydama doğru sıralanmış bir örneğini yazınız.

https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum11

Ek Kaynaklar

- TclButton Kullanımı
  https://www.docs.clomosy.com/TclButton
  https://www.youtube.com/watch?v=W6ZfDO3AJFg

- TclEdit Kullanımı
  https://www.docs.clomosy.com/TclEdit
  https://www.youtube.com/watch?v=vI72lhATzo4&t=145s

- TclLabel Kullanımı
  https://www.docs.clomosy.com/TclLabel
  https://www.youtube.com/watch?v=3fqn6KQpWgE

- TclImage Kullanımı
  https://www.docs.clomosy.com/TclImage
  https://www.youtube.com/watch?v=bSp_8FUS7T0

- TclPanel Kullanımı
  https://www.docs.clomosy.com/TclPanel
  https://www.youtube.com/watch?v=cMf2u80OOD0&t=4s

- TclLayout Kullanımı
  https://www.docs.clomosy.com/TclLayout
  https://www.youtube.com/watch?v=VRO7G-O1Ygc

- TclHorzScrollBox Kullanımı
  https://www.docs.clomosy.com/TclVertScrollBox

- Bileşen Özellikleri
  https://www.docs.clomosy.com/Object_Properties
  https://www.youtube.com/watch?v=oREWkuiYbt4
  https://www.youtube.com/watch?v=neaHs4mCXkM
  https://www.youtube.com/watch?v=wLiQdZkIWaA
  https://www.youtube.com/watch?v=OV5z8SzvUw4&t=18s
  https://www.youtube.com/watch?v=s3MX8w5k-Bg
  https://www.youtube.com/watch?v=Ndu2JuK8l3w
  https://www.youtube.com/watch?v=woC9hgtxBgU&t=17s
  https://www.youtube.com/watch?v=kTAEr2WI_g0&t=4s
  https://www.youtube.com/watch?v=6HHNxMTp3U8&t=1s
  https://www.youtube.com/watch?v=NAfXKxL8sfc
  https://www.youtube.com/watch?v=KHQP5iEs42M&t=3s

BÖLÜM 12 : OLAYLAR (EVENTS)

OLAY NEDİR?

Gerçek hayatta “Olay nedir?” diye sorulduğunda muhtemelen “olan bir şey” diye yanıt verilir. Bu tanım oldukça yerindedir ve programlamada da aynı şekilde geçerlidir. Birçok program, olan etkileşime tepki vererek kullanıcıyı yönlendirir. Örneğin:

- Farenin hareket ettirilmesine veya tıklanmasına,
- Geçen süreye,
- Klavye üzerinde basılan bir tuşa,
- Mobil cihazın ekranına dokunulması gibi.

Programlar içerisindeki birçok fonksiyon genellikle hiçbir şey yapmadan öylece bekler, ta ki birileri onları tetikleyene kadar. Örneğin; bilgisayar açılıp ön yükleme tamamlandığında çoğu zaman bir şey olmaz. Ne zaman ki fare hareket ettirilir ve tıklanmalar yapılır, imleç ekran üzerinde hareket etmeye ya da çeşitli menüler açılmaya başlayıncaya kadar. Bunların gerçekleşmesini sağlayan yapılar *olay (event)* olarak adlandırılır.

Clomosy’de olayları bileşenler üzerinde tanımlayarak formla etkileşime geçilmesi sağlanır. Bunun için *AddNewEvent* özelliği kullanılır. Özellik kullanılırken sırasıyla; olayın ekleneceği bileşen, olayın hangi durumda tetikleneceği ve yapılacak olan işlem belirtilir.

Örnek kullanım aşağıda gösterilmiştir.

```
AnaForm.AddNewEvent(mesajBtn,tbeOnClick,'selamla');
```

Olayın **mesajBtn** isimli bir bileşen üzerinde tanımlanacağını ifade eder.

tbeOnClick ifadesi, yapılacak işin bileşen üzerine tıklandığında tetikleneceğini belirler. Farklı kullanımları da mevcuttur.

Bu kısımda genellikle bir prosedür ya da fonksiyon çağırılır. Burada da **selamla** isimli prosedür çağırılır. Prosedür isminin tırnak içinde belirtildiğine dikkat edilmelidir.

TBEONCLICK

tbeOnClick ifadesini kullanarak bir bileşene tıklama olayı atanabilir. Böylece bileşen üzerine tıklandığında bir işlemin gerçekleştirilmesi sağlanır. Örneğin; bir patlama sesi çıkarma veya oyundaki karakterin zıplaması gibi.

Kod 112'de form üzerine 'MesajBtn' isimli bir buton eklendi ve buton bileşenine AddNewEvent metodu kullanılarak tıklama olayı (tbeOnClick) tanımlandı. Butona tıklandığında 'Selamla' isimli prosedür işleme alınacak ve ekrana "Merhaba" ifadesi yazdırılacaktır.

QR Okut

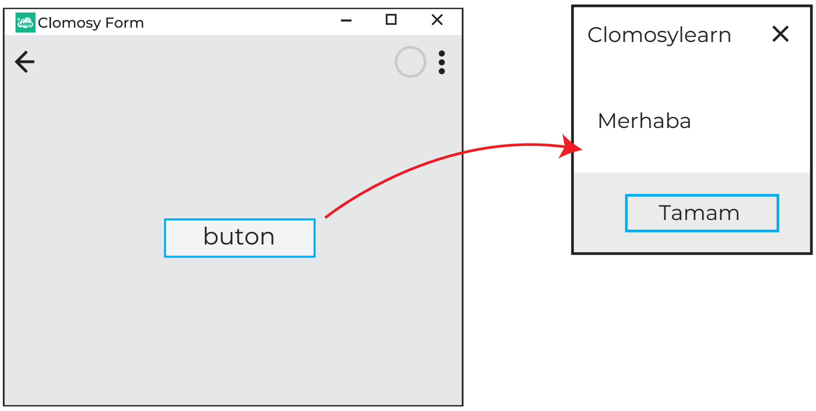
(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum12/Ornek1)

Ana Kod

```
var
 Form1:TclForm;
 MesajBtn:TCLButton;
void Selamla;
{
 ShowMessage('Merhaba');
}
{
 Form1=TclForm.Create(Self);
 MesajBtn=Form1.AddNewButton(Form1,'MesajBtn', 'buton');
 Form1.AddNewEvent(MesajBtn,tbeOnClick,'Selamla');
 Form1.Run;
}
```

Kod 112: tbeOnClick Örneği 1

Çalıştırıldığında *Görsel 102* ekrana gelir.



Görsel 102: tbeOnClick Kullanımı

Kod 113'te form üzerine 'RenklenBtn' isimli bir buton eklendi ve buton bileşenine AddNewEvent metodu kullanılarak tıklama olayı (tbeOnClick) tanımlandı. Butona tıklandığında 'Renklen' isimli prosedür çağrılır ve form arka plan rengi rastgele değişir.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum12/Ornek2)

Ana Kod

```
var
 Form1:TclForm;
 RenkBtn:TCLButton;
 Hex:String;

void Renklen;
var
 Red, Green, Blue: Byte;
{
 Red = clMath.GenerateRandom(0,256);
 Green = clMath.GenerateRandom(0,256);
 Blue = clMath.GenerateRandom(0,256);

 Hex = '#'+ IntToHex(Red,2) + IntToHex(Green,2) + IntToHex(Blue,2);
 Form1.SetFormColor(Hex,'',clGNone);
}

{
 Form1=TclForm.Create(Self);
 RenkBtn=Form1.AddNewButton(Form1,'RenkBtn', 'RENKLEN');
 RenkBtn.Width = 150;
 Form1.AddNewEvent(RenkBtn,tbeOnClick,'Renklen');
 Form1.Run;
}
```

Kod 113: tbeOnClick Örneği 2

clMath.GenerateRandom() Kullanımı:

clMath sınıfına ait olan *GenerateRandom* fonksiyonu kullanılarak başlangıcı ve bitişi belirtilen değerler arasında rastgele bir sayı üretilebilir.
Yukarıdaki örnekte *clMath.GenerateRandom(0,256)* görüldüğü gibi 0 ve 256 arasında rastgele sayı üretilmiştir.

TBEONMOUSEDOWN



tbeOnMouseDown olayını kullanarak bir bileşene mouse (fare) tuşuna basma olayı tanımlanabilir. Bu olayın tıklamadan farkı, tuşa basıldığı anda çalışmasıdır; parmağın tuşu serbest bırakmasına gerek duymaz. Örneğin; masaüstündeki bir simgeyi sürüklerken simge üzerine gelip mouse sol tuşuna basıldığında simgeyi tutmak gibi.

Kod 114'te form üzerine AddNewEvent metodunu kullanılarak bir mouse tuşuna basma olayı (tbeOnMouseDown) tanımlandı. Mouse tuşuna basıldığında 'MesajYaz' prosedürü işleme alınarak ekrana "tbeOnMouseDown çalıştı" mesajı gelecektir.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum12/Ornek3)

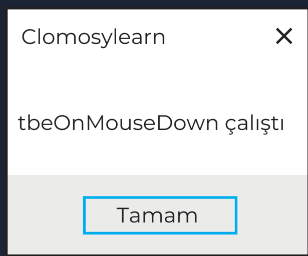
Ana Kod

```
var
 Form1:TclForm;

void MesajYaz;
{
 ShowMessage('tbeOnMouseDown çalıştı');
}

{
 Form1 = TclForm.Create(Self);
 Form1.AddNewEvent(Form1,tbeOnMouseDown,'MesajYaz');

 Form1.Run;
}
```



Kod 114: tbeOnMouseDown Örneği

Tuşa basılı tutarak tuşun serbest bırakılmasına gerek kalmadan ilk basıldığında çalıştığı anlaşılabilir.

TBEONMOUSEUP



tbeOnMouseUp ifadesini kullanarak bir bileşene, mouse tuşunu basılı halden tekrar serbest hâle getirme olayı tanımlanabilir. Bu olayın tıklamadan farkı; tuşa basıldığında ya da basılı tutulduğunda etkilenmemesi, tuşun serbest bırakılması ile tetiklenmesidir. Masa üstündeki simgeyi sürüklerken mouse tuşu serbest bırakıldığında simgenin penceredeki konumunu alması gibi.

Kod 115'te form üzerine AddNewEvent metodu kullanılarak bir Mouse tuşunu serbest bırakma olayı (tbeOnMouseUp) tanımlandı. Mouse tuşu serbest bırakıldığında 'MesajYaz' prosedürü işleme alınarak ekrana "tbeOnMouseUp çalıştı" mesajı gelecektir.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum12/Ornek4)

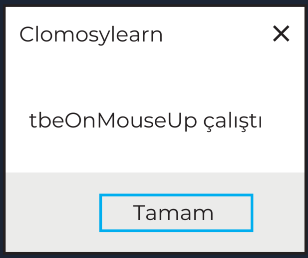
Ana Kod

```
var
  Form1:TclForm;

void MesajYaz;
{
  ShowMessage('tbeOnMouseUp çalıştı');
}

{
  Form1 = TclForm.Create(Self);
  Form1.AddNewEvent(Form1,tbeOnMouseUp,'MesajYaz');

  Form1.Run;
}
```



Kod 115: tbeOnMouseUp Örneği

Tuşu bir süre basılı tuttuktan sonra serbest bırakıldığında olayın çalıştığı anlaşılır.

Kod 116'da form üzerine AddNewEvent özelliği kullanılarak hem mouse tuşuna basma olayı (tbeOnMouseDown) hem de mouse tuşunu serbest bırakma olayı (tbeOnMouseUp) tanımlandı. Mouse tuşuna basıldığında ve tuş serbest bırakıldığında ekranda görünen yazı ve arka plandaki renk geçişi değişecektir.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum12/Ornek5)

Ana Kod

```
var
 Form1:TclForm;
 SonDurum:TCLLabel;

void MouseBas;
{
 Form1.SetFormColor('ffffff','#00b282',clGCross);
 SonDurum.Caption = ' Tuşa bastın';
}

void MouseBirak;
{
 Form1.SetFormColor('#00b282','ffffff',clGCross);
 SonDurum.Caption = ' Tuşu serbest bıraktın';
}

{
 Form1=TclForm.Create(Self);

 SonDurum = Form1.AddNewLabel(Form1,'SonDurum','Henüz Tıklamadın');
 SonDurum.Align = alContents;
 SonDurum.StyledSettings = ssFamily;
 SonDurum.TextSettings.Font.Size=20;
 SonDurum.margins.left=10;

 Form1.AddNewEvent(Form1,tbeOnmouseUp,'MouseBirak');
 Form1.AddNewEvent(Form1,tbeOnmouseDown,'MouseBas');

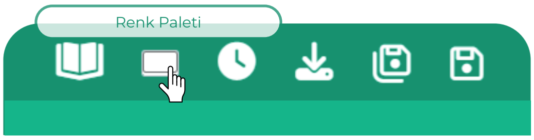
 Form1.Run;
}
```

Kod 116: tbeOnMouseUp – tbeOnMouseDown Örneği

TBEONMOUSEMOVE

tbeOnMouseMove ifadesini kullanarak imleç bir bileşenin üzerine getirildiğinde tetiklenecek bir olay tanımlanabilir.

Örneğin: *Görsel 103'*teki *Clomosy Geliştirme Ortamında* imleç, *Renk Paleti* üzerine getirildiğinde ögenin isminin ekranda görünmesi gibi.



Görsel 103: Clomosy Geliştirme Ortamı Renk Paleti Gösterimi

*Kod 117'*de bir form ve panel oluşturuldu. Bu panelin içine bir resim eklendi ve fare hareketlerine tepki vererek resmi konumlandıran bir prosedür tanımlandı. İşlem adımları şu şekildedir:

1.Form ve Panel Oluşturma: Form oluşturulduğunda formun alt kısmına hizalanmış bir panel eklenir.

2.Görsel Ekleme: Panelin içine bir resim yerleştirilir. Resim, panelin yüksekliğine eşit olacak şekilde boyutlandırılır ve panelin altına hizalanır.

3.Görsel Döndürme: Resim, 180 derece döndürülür (RotationAngle özelliği kullanılarak).

4.Fare Hareketlerine Tepki Verme: Fare panel üzerinde hareket ettikçe 'HareketEttir' prosedürü çalışır. Bu prosedürde, resmin X konumu, fare imlecinin X konumuna *(clSenderMousePosX)*, Y konumu ise fare imlecinin Y konumunun yarısına *(clSenderMousePosY)* ayarlanır. Bu şekilde, fare panelin üzerinde hareket ettikçe resim de panelin içinde hareket eder.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum12/Ornek6)

Ana Kod

```
var
  Form1:TclForm;
  Panel1 : TclPanel;
  Image1 :TCLImage;
  url : String;
void HareketEttir;
{
  Image1.Position.X = Form1.clSenderMousePosX;
  Image1.Position.Y = Form1.clSenderMousePosY / 2;
}

{
  Form1 = TclForm.Create(Self);
  Panel1 = Form1.AddNewPanel(Form1,'Panel1');
  Panel1.Align = alBottom;
  Panel1.Height = 100;

  Image1 = Form1.AddNewImage(Panel1,'Image1');
  url = 'https://clomosy.com/WildRun/arrowButton_Mobile.png';
  Form1.SetImage(Image1, url);
  Image1.Align = alNone;
  Image1.Height = Panel1.Height;
  Image1.Width = Panel1.Height;
  Image1.Position.X = 0;
  Image1.Position.Y = Panel1.Height - (Image1.Height);
  Image1.RotationAngle = 180;

  Form1.AddNewEvent(Panel1,tbeOnMouseMove,'HareketEttir');

  Form1.Run;
}
```

Kod 117: tbeOnMouseMove Örneği

BİLGİLENDİRME:

Yukarıdaki örneğin uygulamasına aşağıdaki link üzerinden erişim sağlayabilirsiniz.
https://www.youtube.com/watch?v=grPFEUSaHp0

MOUSE KONUMUNU YAKALAMA

Program içerisinde mouse imlecinin konumunu almak için *clSenderMousePosX* ve *clSenderMousePosY* özellikleri kullanılır. Bu özellikler aşağıdaki şekilde işlev görür:

- **clSenderMousePosX:** Mouse imlecinin yatay (X) konumunu verir.
- **clSenderMousePosY:** Mouse imlecinin dikey (Y) konumunu verir.

Kod 118'de, görselin sol üst köşesi yatay ve dikey konum için sıfır noktası olarak kabul edilir. Mouse imlecinin görsel üzerindeki hareketi bu konumlara göre hesaplanır ve ekrana yazdırılır. Bu sayede imlecin, görsel üzerindeki hareketini yatay ve dikey konum olarak takip edilebilir.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum12/Ornek7)

Ana Kod

```
var
 Form1:TclForm;
 KonumLbl:TclLabel;
 HareketImg:TClImage;

void MouseHareket;
{
 KonumLbl.Text = IntToStr(Form1.clSenderMousePosX) + ','+IntToStr(Form1.clSenderMousePosY);
}

{
 Form1 = TClForm.Create(Self);

 KonumLbl = Form1.AddNewLabel(Form1, 'KonumLbl','--');
 KonumLbl.Align = alTop;
 KonumLbl.Height = 20;

 HareketImg = Form1.AddNewImage(Form1, 'HareketImg');
 Form1.setImage(HareketImg,'https://clomosy.com/demos/bg.png');
 HareketImg.Align = alCenter;
 HareketImg.Height = 260;
 HareketImg.Width = 145;
 Form1.AddNewEvent(HareketImg, tbeOnMouseMove,'MouseHareket');

 Form1.Run;
}
```

Kod 118: clSenderMousePosX – clSenderMousePosY Örneği

TBEONENTER

tbeOnEnter ifadesini kullanarak bir bileşene odaklanıldığında yani bileşen klavye üzerinden veya tıklama yoluyla seçildiğinde, tetiklenecek bir olay tanımlanabilir. tbeOnEnter olayı, kullanıcı arayüzünde odaklanabilen hemen hemen tüm bileşenlerde kullanılabilir.

Kod 119'da form üzerine iki buton eklendi ve butonlara AddNewEvent metodu kullanılarak tbeOnEnter olayı tanımlandı. Butonlardan birisi seçildiğinde ilgili prosedür çalışacak ve form seçilen takımın rengini alacaktır.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum12/Ornek8)

Ana Kod

```
var
  Form1:TclForm;
  ButtonGS,ButtonFB:TclButton;
  Takim:String;
void RenkVerFB;
{
  Form1.SetFormColor('#ffef03','#004684',clGCross);
}
void RenkVerGS;
{
  Form1.SetFormColor('#a90432','#fdb912',clGCross);
}
{
  Form1 = TclForm.Create(Self);

  ButtonGS = Form1.AddNewButton(Form1, 'ButtonGS', 'Galatasaray');
  With ButtonGS do
  {
    Align = alRight;
    Margins.Right = 10;
    Margins.Bottom = 10;
    Width = 100;
  }
  Form1.AddNewEvent(ButtonGS,tbeOnEnter,'RenkVerGS');

  ButtonFB = Form1.AddNewButton(Form1,'ButtonFB','Fenerbahçe');
  With ButtonFB do
  {
    Align = alLeft;
    Margins.Left = 10;
    Margins.Bottom = 10;
    Width = 100;
  }
  Form1.AddNewEvent(ButtonFB,tbeOnEnter,'RenkVerFB');
  Form1.Run;
}
```

Kod 118: clSenderMousePosX – clSenderMousePosY Örneği

TBEONCHANGE

tbeOnChange ifadesi kullanılarak bileşenin içeriği değiştiğinde tetiklenen bir olay tanımlanabilir. Genellikle kullanıcı tarafından değiştirilebilen bileşenlerde kullanılır. TclEdit, TclMemo, TclCombobox gibi bileşenler örnek verilebilir.

Kod 120'de bir form oluşturuldu ve üzerine bir TclComboBox nesnesi eklendi. Kullanıcı bu nesneden bir gün seçtiğinde seçilen günü bir mesaj kutusunda gösterir.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum12/Ornek9)

Ana Kod

```
var
 Form1:TclForm;
 ComboBox1 : TClComboBox;

void ComboSecilenDeger;
{
  ShowMessage('Seçilen gün : '+ ComboBox1.Items[ComboBox1.ItemIndex]);
}

{
 Form1 = TclForm.Create(Self);
 ComboBox1 = Form1.AddNewComboBox(Form1,'ComboBox1');
 With ComboBox1 do
 {
   Align = alTop;
   Width = 150;
   Margins.Top = 10;
   Margins.Left = 50;
   Margins.Right = 50;
   AddItem('Pazartesi','01');
   AddItem('Salı','02');
   AddItem('Çarşamba','03');
   AddItem('Perşembe','04');
   AddItem('Cuma','05');
   AddItem('Cumartesi','06');
   AddItem('Pazar','07');
 }

 Form1.AddNewEvent(ComboBox1,tbeOnChange,'ComboSecilenDeger');
 Form1.Run;
}
```

Kod 120: tbeOnChange Örneği

TclComboBox Nedir?

Önceden tanımlanmış listeden seçim yapmasına olanak tanıyan bir bileşendir.
Ayrıntılı bilgi almak için bölümün sonundaki "Ek Kaynaklar" kısmından dokümanlara giderek incelenebilir.

KLAVYE GİRİŞİNİ YAKALAMAK

Kullanıcı klavyede bir tuşa bastığında veya yazı yazdığında bu durumu algılayıp belirli işlemler yapabilmeyi sağlar. Bu, genellikle *tbeOnFormKeyDown, tbeOnFormKeyUp* olaylarıyla gerçekleştirilir. Örneğin; bir metin kutusuna yazılan her harfi kontrol edebilir, sadece belirli tuşlara izin verebilir veya bir tuşa basıldığında başka bir işlem tetiklenebilir. Bu şekilde, kullanıcıların uygulama içindeki etkileşimleri daha kontrollü ve dinamik hâle getirilebilir.

Clomosy'de *clSenderKeyChar,* bir klavye olayı işlendiğinde bu olayı tetikleyen bileşeni temsil eder. Bu değer, kullanıcının klavyeden girdiği karakteri veya tuşun karakter olduğunu ifade eder. Bu olaydan geriye bir değer alabilmek için tbeOnFormKeyUp ve tbeOnFormKeyDown olay tipleri kullanılır.

clSenderChar klavyeden alınan girdi bilgisine göre işlem yapabilmek için tercih edilir. Bu özellik ile klavyeden basılan ya da çekilen tuş bilgisi decimal değerde tutulur. Eğer klavye tuşlarının decimal karşılığı bilinmiyorsa 'Key Code Table' internette aranarak decimal kodları bulunabilir.

Öncelikle tbeOnFormKeyDown ve tbeOnFormKeyUp olaylarının neler olduğuna bakalım.

TBEONFORMKEYDOWN

tbeOnFormKeyDown ifadesi tbeOnMouseDown gibi davranır. tbeOnFormKeyDown olay tipini kullanarak klavye üzerindeki bir tuşa basma olayı tanımlanır. Olay, klavye üzerindeki bir tuşa basıldığında çalışır, parmağın tuşu serbest bırakmasına gerek duymaz.



TBEONFORMKEYUP

tbeOnFormKeyUp olay tipi tbeOnMouseUp gibi davranır. tbeOnFormKeyUp ifadesini kullanarak klavye üzerinde basılı haldeki bir tuşun, tekrar serbest bırakılması olayı tetiklenir. Olay, klavye üzerinde basılı haldeki bir tuşun serbest bırakılmasıyla çalışır.

Örnekte 'boşluk' ve 'w' tuşlarına birer tbeOnFormKeyDown, 'a' ve 'd' tuşlarına birer tbeOnFormKeyUp olayı tanımlandı. Uygulamada basılan tuşa göre gerçekleşecek olay ile hangi tuşa basıldığı tespit edilerek basılan tuşla ilgili bir mesaj gösterildi *(bkz. Kod 121).*

Bu uygulama kodları Windows üzerinde çalıştırmalıdır.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum12/Ornek10)

Ana Kod

```
var
 Form1:TCLForm;

void KeyDown
{
 if (Form1.clSenderKeyChar == 32)
 {
  ShowMessage('Space Tuşuna Basıldı');
 }
 else if (Form1.clSenderKeyChar == 119)
 {
  ShowMessage('w Tuşuna Basıldı');
 }
}

void KeyUp
{
 if (Form1.clSenderKeyChar == 97)
 {
  ShowMessage('a Tuşundan Çekildi');
 }
 else if (Form1.clSenderKeyChar == 100)
 {
  ShowMessage('d Tuşundan Çekildi');
 }
}

{
 Form1 = TCLForm.Create(Self);
 Form1.AddNewEvent(Form1, tbeOnFormKeyDown, 'KeyDown');
 //Form1.AddNewEvent(Form1, tbeOnFormKeyUp, 'KeyUp');
 Form1.Run;
}
```

Kod 121: tbeOnFormKeyDown – tbeOnFormKeyUp Örneği

TBEONVIRTUALKEYBOARDSHOWN VE TBEONVIRTUALKEYBOARDHIDDEN

VirtualKeyboard özelliği, mobil uygulamaların sanal klavyeyi kontrol etmesine olanak tanır. Bu özellik, mobil cihazlarda sanal klavyeyi gösterme, gizleme veya klavyenin durumunu izleme gibi işlemleri gerçekleştirmek için kullanılır. Formun boyutu otomatik olarak ayarlanabilir veya içerik, klavye tarafından kapatılmaması için kaydırılabilir. Mobil uygulamalarda kullanıcı deneyimini iyileştirmek ve kullanılabilirliği artırmak için oldukça önemlidir. Klavye olaylarını algılayarak klavye durumuna göre uygulama içindeki etkileşimlerin ayarlanması ve buna uygun olarak yanıt vermesi mümkün hâle gelir.

Bir formda, birden fazla özellik sanal klavye olaylarıyla birlikte kullanılabilir. Sanal klavye olaylarını ele alırken sanal klavyenin konumu ve boyutu hakkında bilgi almak için *Tablo 17*'deki özelliklerden yararlanılır.

Özellik	Açıklama
clVKBoundsWidth	Klavye genişlik değerini verir
clVKBoundsHeight	Klavye yükseklik değerini verir
clVKBoundsRight	Klavyenin sınır sağ kısmının konumunu verir
clVKBoundsLeft	Klavyenin sınır sol kısmının konumunu verir

Tablo 17: VirtualKeyboard Özellikleri

Örnek kullanımı;

```
ShowMessage(Form1.clVKBoundsLeft);
```

Sanal klavye, uygulama ekranında belirdiğinde *tbeOnVirtualKeyboardShown* olayı tetiklenir. Kullanıcı, bir metin girişi alanına (TclEdit veya TclMemo vb.) tıkladığında sanal klavye açılır ve bu olay etkin hâle gelir. Sanal klavye, uygulama ekranında gizlendiğinde *tbeOnVirtualKeyboardHidden* olayı tetiklenir. Kullanıcı, metin girişi alanından çıktığında veya sanal klavyeyi kapattığında bu olay etkin hâle gelir.

Kod 122'de tbeOnVirtualKeyboardShown ve tbeOnVirtualKeyboardHidden olayları örneklendirildi. Metin yazmak için tıklandığında tbeOnVirtualKeyboardShown olayı devreye girer, klavye açıldığında metin alanı ve butonun altındaki boşluk "Layout1.Margins.Bottom = Form1.clVKBoundsHeight;" ifadesi ile değiştirilip bileşenlerin klavyenin altında kalmaları engellenir. Butona tıklandığında ise tbeOnVirtualKeyboardHidden olayı ile ekran başlangıç durumuna döner.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum12/Ornek11)

Ana Kod

```
var
  Form1 : TclForm;
  EditYaz: TclEdit;
  ButonKapat : TclButton;
  Layout1 : TclLayout;

void KlavyeGoster;
{
  Layout1.Margins.Bottom = Form1.clVKBoundsHeight;
}
void KlavyeGizle;
{
  Layout1.Margins.Bottom = 20;
}

{
  Form1 = TclForm.Create(Self);
  Form1.AddNewEvent(Form1,tbeOnVirtualKeyboardShown,'KlavyeGoster');
  Form1.AddNewEvent(Form1,tbeOnVirtualKeyboardHidden,'KlavyeGizle');

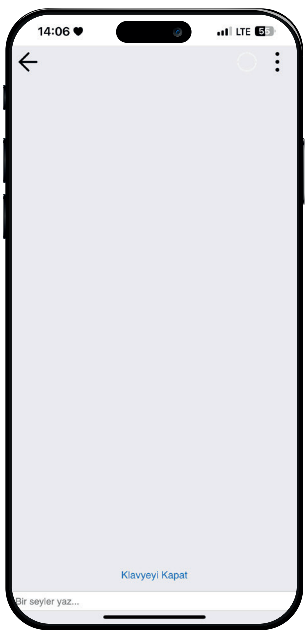
  Layout1 = Form1.AddNewLayout(Form1,'Layout1');
  Layout1.Align = alClient;
  Layout1.Margins.Bottom = 20;

  EditYaz = Form1.AddNewEdit(Layout1,'EditYaz','Bir şeyler yaz...');
  EditYaz.Align = AlMostBottom;
  EditYaz.Height = 50;

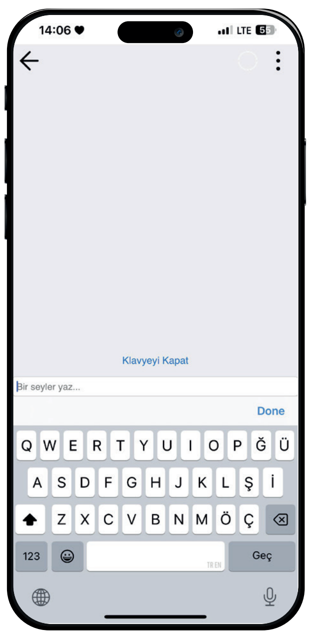
  ButonKapat=Form1.AddNewButton(Layout1,'ButonKapat','Klavyeyi Kapat');
  ButonKapat.Align = alBottom;
  ButonKapat.Margins.Bottom = 10;
  ButonKapat.Margins.Top = 10;

  Form1.Run;
}
```

Kod 122: tbeOnVirtualKeyboardShown – tbeOnVirtualKeyboardHidden Örneği



Görsel 104: tbeOnVirtualKeyboardHidden Kullanımı



Görsel 105: tbeOnVirtualKeyboardShown Kullanımı

Alıştırmalar

1. Olay nedir ve uygulamalarda kullanıcı etkileşimini nasıl etkiler?

2. Fare ve klavye olayları arasındaki farklar ve benzerlikler nelerdir? Fare ve klavye olaylarına birer örnek veriniz.

3. Sanal bir klavye kullanırken hangi olaylar meydana gelebilir? Bu olaylar bir mobil uygulamayı nasıl etkileyebilir?

4. Bir buton oluşturularak butona tıklama olayı tanımlayınız. Butona tıklandığında formun arka planına görsel ekleyiniz.

https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum12

Ek Kaynaklar

- Olaylar (Events) Kullanımı
  https://www.docs.clomosy.com/Object_Properties#TCLBasisEvent

- Sanal Klavye Kullanımı
  https://www.docs.clomosy.com/Virtual_Keyboard

- TclComboBox Kullanımı
  https://www.docs.clomosy.com/TclComboBox
  https://www.youtube.com/watch?v=zxI45dAaKpA

BÖLÜM 13: BİRİMLER (UNITS)

BİRİM (UNIT) NEDİR?

Birim *(Unit)*, programın fonksiyon ve özelliklerinin, bölümlere ayrılmasını, sadeleştirilmesini ve yönetilmesini kolaylaştırmak için kullanılır. Birim, kodun modüler hale getirilmesine ve farklı bölümlerin birbirinden bağımsız olarak organize edilmesine olanak tanır.

Birimler; kodun okunabilirliğini artırır, bakımını kolaylaştırır ve projelerin yeniden kullanılabilir parçalar halinde organize edilmesini sağlar.



Örneğin, program büyük bir eve benzetilecek olursa; Bu evin içinde farklı odalar (birimler) bulunur ve her oda farklı bir amaca hizmet eder. Buna göre:

- Mutfak: Yemek yapmak için
- Oturma Odası: Dinlenmek ve misafir ağırlamak için
- Banyo: Temizlik ve hijyen için
- Yatak Odası: Uyuma ve dinlenme için kullanılır.

Evdeki her oda (birim), belirli bir görevi yerine getirir ve bu odalar arasında bir düzen vardır. Her bir odanın ayrı bir işlevi olmasına rağmen hepsi birlikte çalışarak evin bir bütün olarak görevini yerine getirmesini sağlar.

Bir yemek yapma uygulaması yazılacağını varsayalım. Bu uygulama, 3 ayrı bölüme ayrılabilir:

Tarifleri Yönetme Birimi (Mutfak):

- Bu birim, tarifleri eklemek, düzenlemek ve silmek gibi işlemlerle ilgilenir.
- İçinde ‘TarifEkle’, ‘TarifSil’ gibi prosedürler bulunur.

Alışveriş Listesi Birimi (Market Listesi):

- Bu birim, tariflerde kullanılan malzemelerden alışveriş listesi oluşturmak için kullanılır.
- İçinde 'ListeyeMalzemeEkle', 'ListeyiGoruntule' gibi prosedürler yer alır.

Kullanıcı Arayüzü Birimi (Rehber- Ana Kod):

- Bu birim, uygulamanın görsel kısmıyla yani kullanıcıya gösterilecek merkezi yönetim yeridir.
- İçinde kullanıcıdan tarif almak veya listeyi göstermek için kodlar bulunur.

Her bir birim (Mutfak, Market Listesi, Rehber) kendi işini yapar ama hepsi bir arada çalışarak yemek yapma uygulamasını tam anlamıyla işlevsel hale getirir. Birimlerin bu şekilde kullanılması; kodun daha anlaşılır, yönetilebilir ve bakımın kolay olmasını sağlar.

Birim (Unit) Kullanımı Neden Önemlidir?

Kodun Modülerliği: Birimler, kodun daha düzenli ve modüler olmasını sağlar. Her bir birim, programın farklı bir işlevselliğini temsil eder.

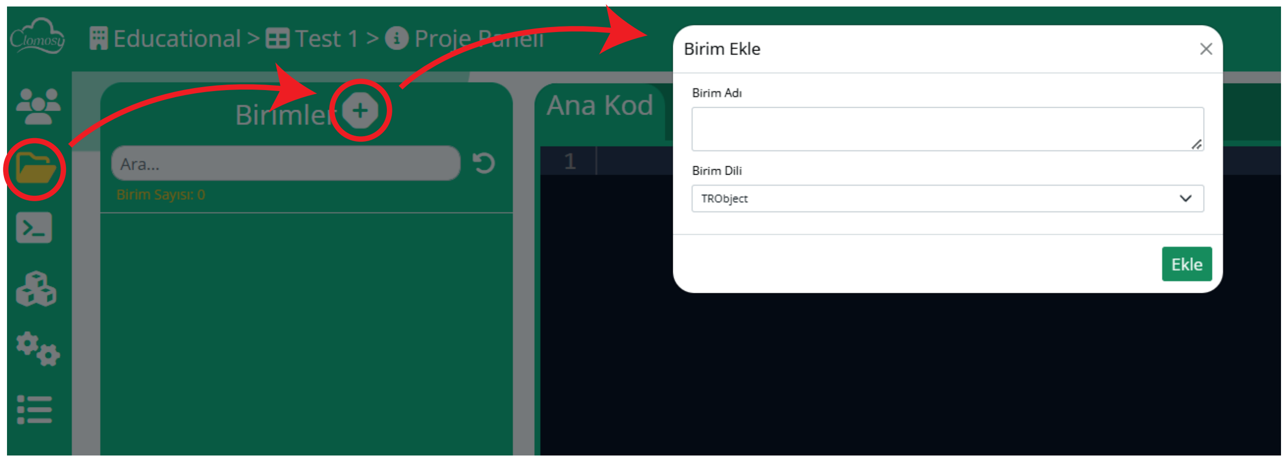
Yeniden Kullanılabilirlik: Bir birim, bir kez yazıldıktan sonra başka projelerde de kullanılabilir. Bu, kod tekrarını azaltır ve yazılım geliştirme sürecini hızlandırır.

Bakım Kolaylığı: Birimler, kodun bakımını ve yönetimini kolaylaştırır. Büyük projelerde, belirli bir işlevdeki hatayı bulmak ve düzeltmek daha kolay hale gelir.

BİRİM (UNIT) OLUŞTURMA

Birim oluşturabilmek için *Görsel 104*'teki gibi kod editöründe, sol menüde bulunan dosya ikonuna tıklanmalıdır.

Bu kısımda "Birimler" bölümü açılır. Birim oluşturabilmek için artı (+) simgesine tıklandığında birim ekleme kutucuğu ekrana gelir. Açılan "Birim Ekle" kutucuğunda 'Birim Adı' ve 'Birim Dili' girilebilecek alanlar yer alır.



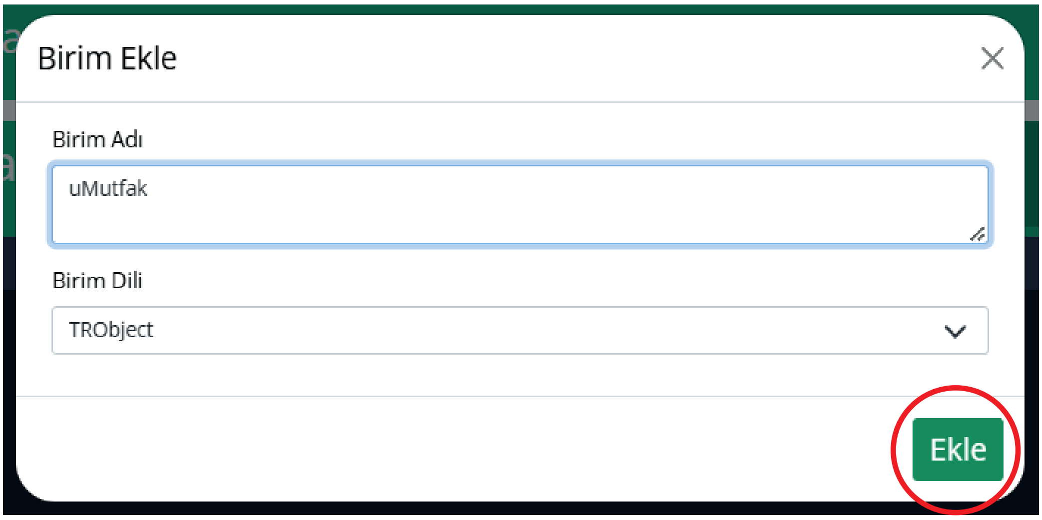
Görsel 106: Birim (Unit) Oluşturma Adımları

Birim (Unit) İsimlendirme:

Birim isimlendirme kurallarına dikkat etmek, kodlama sürecinde işleri daha düzenli ve anlaşılır hale getirir. Aşağıdaki noktalar bu konuda bize yardımcı olur:

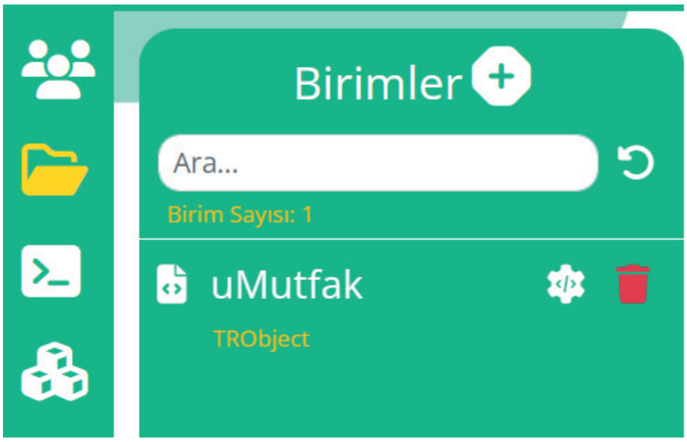
- **Türkçe karakter kullanmayın:** Birim isimlerinde Türkçe karakterler (ç, ı, ö, ş, ü) kullanmaktan kaçının. İngilizce karakterler kullanarak isimlerinizi yazmak, farklı sistemlerde uyumluluğu artırır ve kodunuzu daha taşınabilir hale getirir. Birim isimleride kodlama içindeki değişken isimleri kurallarına tabidir ve kod içinde kullanılırlar.
- **Kelimeler arasında boşluk kullanmayın:** Birim isimlerinde kelimeleri ayırmak için boşluk yerine birleşik yazım veya alt çizgi (_) kullanılabilir. Bu, isimlerin daha okunabilir ve yönetilebilir olmasını sağlar.
- **Standart sağlamak için aynı harf ile başlatın. Örneğin önüne 'u' harfi ekleyin:** Birim isminin başına 'u' harfi eklemek, kodlama esnasında bu yapının bir birim olduğunu hatırlatır. Örneğin, uEvModeli (veya uEv_Modeli) gibi bir isim, bu yapının birim olduğunu gösterir ve diğer yapılarla karışmasını önler.

'Birim Adı' alanına, örnekte "Tarifleri Yönetme (Mutfak)" birimi olan "uMutfak" birim ismi yazılır. TRObject dili ile kodlama yapılacağı için 'Birim Dili' TRObject olarak kalır. Son olarak sağ alt kısımda bulunan 'Ekle' butonuna tıklayarak birim oluşturulur *(bkz. Görsel 105).*



Görsel 107: Birim (Unit) Ekleme Örneği

'Ekle' butonuna tıklandığında "Birimler" bölümündeki listeye eklenir *(bkz. Görsel 106).*



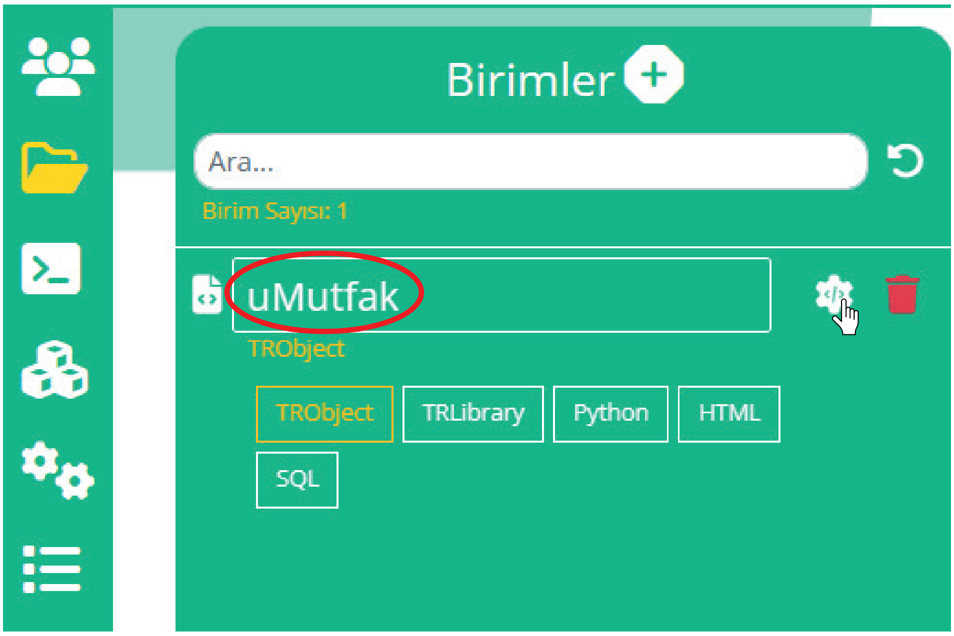
Görsel 108: Birimler (Units) Listesi

Birim oluşturuldu.

BİRİM (UNIT) ADI DEĞİŞTİRME VE SİLME İŞLEMLERİ

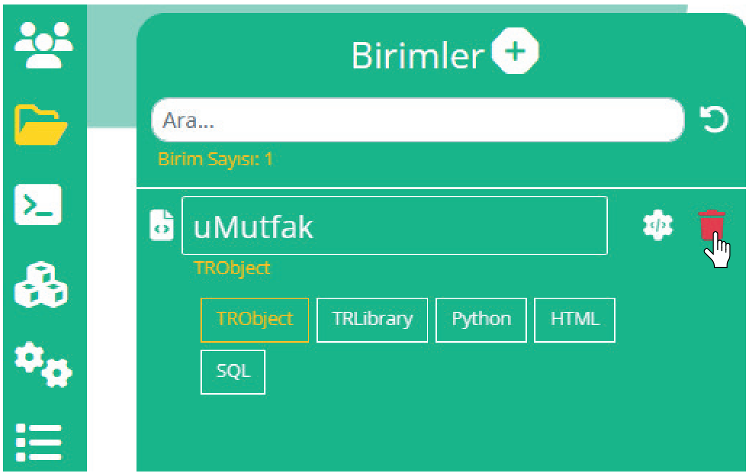
Proje içerisinde oluşturulan birimler, silinebilir veya isimleri değiştirilebilir. Bu işlemleri gerçekleştirmek için aşağıdaki adımlar takip edilmelidir:

- Yan menüden "Birimler" sekmesine tıklanmalıdır.
- İsmi değiştirilmek istenen birimin, yanında bulunan ayarlama butonuna tıklayarak açılan kutu üzerinde, birimin adı değiştirilebilir ve birim dili düzenlenebilir*(bkz. Görsel 107)*.
- Birimi silmek için aynı ekranda bulunan çöp kutusu simgesine tıklanmalıdır*(bkz. Görsel 108)*.



Görsel 109: Birim (Unit) Adı Değiştirme

DİKKAT:
Silinmek istenen birim, sekmede açık ise silme işlemi gerçekleştirilemez. Birimi silebilmek için öncelikle bu birim sekmede kapatılmalıdır.



Görsel 110: Birim (Unit) Silme

Bu uygulamada “Ana Kod” içerisine bir form ve buton eklendi. Butona tıklandığında oluşturulan birime (uMutfak) gider. Bu birim içerisinde ise TclLabel nesnesi eklendi ve içerisine birimin adı yazıldı.

Öncelikle “Ana Kod” içerisine aşağıdaki kodlar yazılmalı ve kaydedilmelidir *(bkz. Kod 123).*

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum13/Ornek1)

Ana Kod

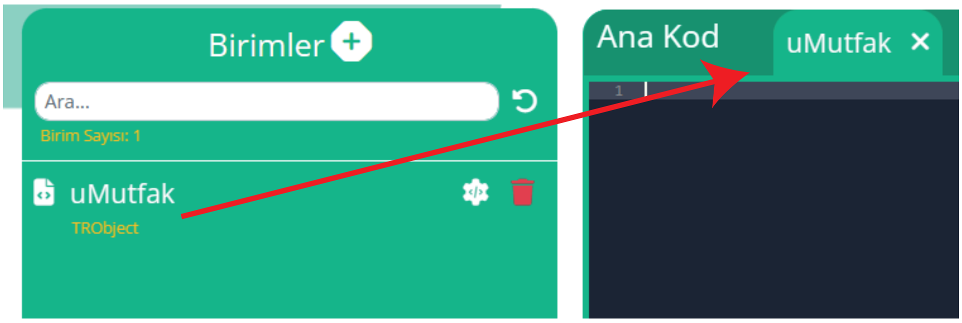
```
var
 Form1:TCLForm;
 Button1 : TclButton;
/*
void BirimeGit;
{

}
*/
{
 Form1 = TCLForm.Create(Self);
 Button1 = Form1.AddNewButton(Form1,'Button1', 'Birim''e git!');
 Button1.Width = 100;
 //Form1.AddNewEvent(Button1,tbeOnClick,'BirimeGit');
 Form1.Run;
}
```

Kod 123: Ana Kod Ekranı 1

Burada dikkat edilmesi gereken nokta; butonun tıklanma olayı ve yönlendirileceği metot (void) yorum satırları içerisine alınmıştır.

Şimdi sırada, birim içerisine TclLabel nesnesi eklemek var. Bunun için "Birimler" bölümünde oluşturulan birim üzerine tıklanmalıdır. Böylece, ilgili birim (uMutfak), yeni sekmede açılacaktır *(bkz. Görsel 109).*

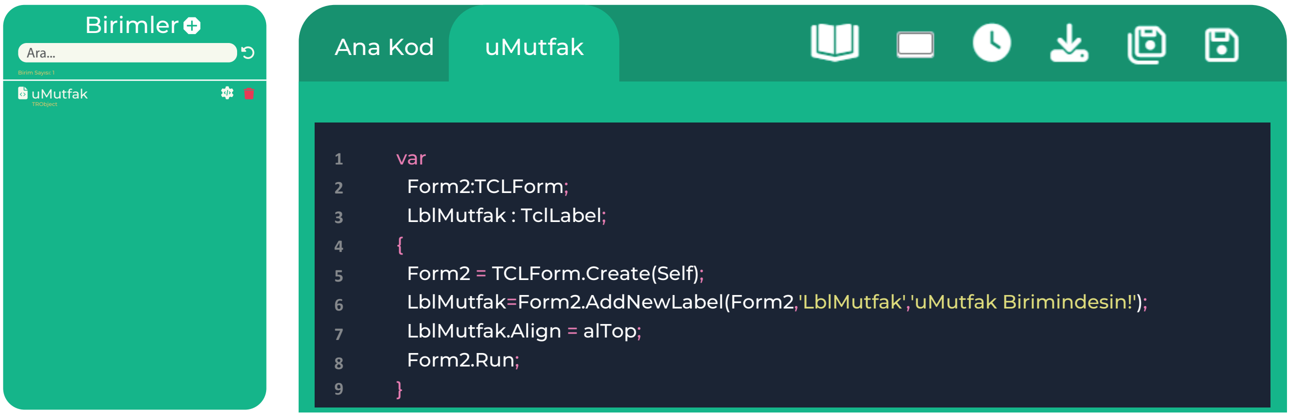


Görsel 111: Birim (Unit) Kullanımı

Açılan bu sekme üzerinde TclLabel nesnesi eklenerek tasarıma devam edilir *(bkz. Kod 124).*

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum13/Ornek2)




```
var
  Form2:TCLForm;
  LblMutfak : TclLabel;
{
  Form2 = TCLForm.Create(Self);
  LblMutfak=Form2.AddNewLabel(Form2,'LblMutfak','uMutfak Birimindesin!');
  LblMutfak.Align = alTop;
  Form2.Run;
}
```

Kod 124: uMutfak Birimi 1

Hem "Ana Kod" hem de birim (uMutfak) sayfaları oluşturulduğuna göre artık yönlendirme işlemlerine başlanabilir. Bu yönlendirme, "Ana Kod" sayfasından belirli bir olaya yanıt olarak "uMutfak" birimine geçiş yapılmasını sağlar.

Bir birimi çağırmak için iki yöntem bulunur. Bu yöntemlerden biri; sadece birimi çağırmaya odaklanırken *(RunUnit)*, diğeri hem birimi çağırır hem de çağıran birim içerisindeki nesnelere erişim sağlar *(TclUnit)*. Aşağıda bu iki yöntem açıklanmaktadır.

BİRİM (UNIT) ÇAĞIRMA YÖNTEMLERİ

RUNUNIT

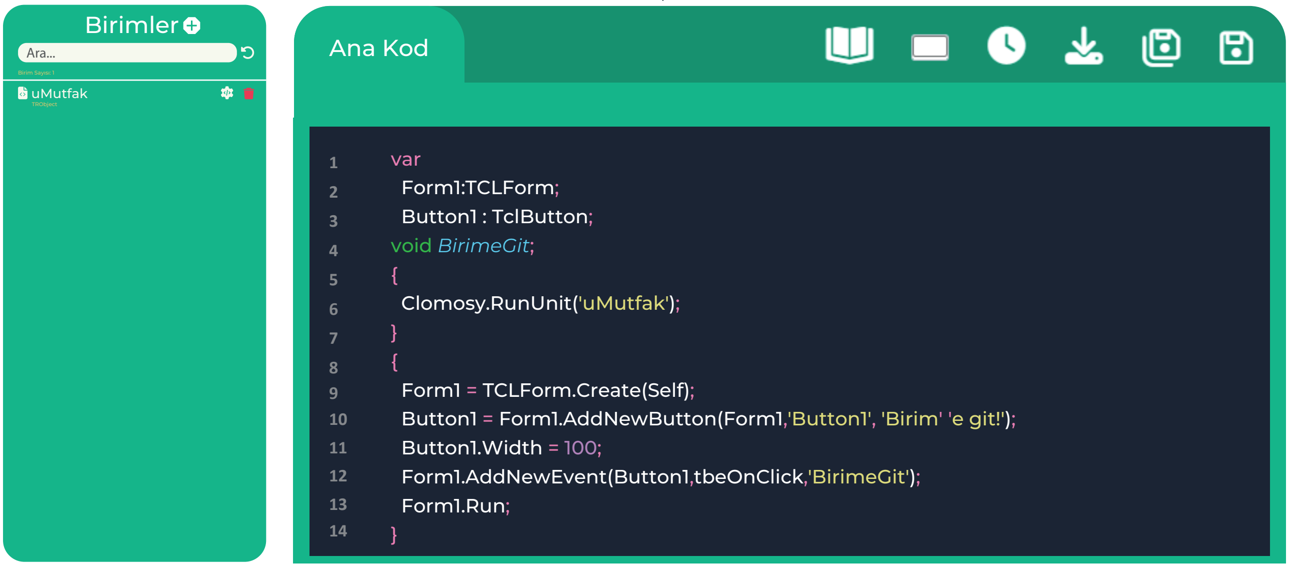
Oluşturulan "uMutfak" birimini istenilen sayfada çağırmak için kullanılır. Clomosy'de standart birim çağırma yöntemidir. Birim adını içeren bir parametre alır. Kullanımı aşağıdaki gibidir:

```
Clomosy.RunUnit('uMutfak');
```

Örnekte; "Ana Kod" içerisinde butona tıklandığında birim çağırmak istendi. Öncesinde yorum satırına alınan kod satırları, tekrar aktif hale getirilir. Ardından, "birimeGit" prosedürü içerisinde birimi çağırma işlemi gerçekleşir *(bkz. Kod 125).*

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum13/Ornek3)




```
var
  Form1:TCLForm;
  Button1 : TclButton;
void BirimeGit;
{
  Clomosy.RunUnit('uMutfak');
}
{
  Form1 = TCLForm.Create(Self);
  Button1 = Form1.AddNewButton(Form1,'Button1', 'Birim' 'e git!');
  Button1.Width = 100;
  Form1.AddNewEvent(Button1,tbeOnClick,'BirimeGit');
  Form1.Run;
}
```

Kod 125: RunUnit Örneği

Kod 125 çalıştırıldığında ve butona basıldığında oluşturulan “uMutfak” birimine gidilir. Birimde yazılan kodlar çalıştırılır. Bu, “uMutfak” birimindeki bileşenlerin ve verilerin, butona tıklama olayı sonucu ekranda görünmesini sağlar.

Son işlem olarak, “uMutfak” biriminde bir buton oluşturuldu. Bu butona tıklandığında sayfanın kapanması ve bir önceki sayfaya dönülmesi sağlanacaktır *(bkz. Kod 126).*

BİLGİLENDİRME:
Sadece “Ana Kod” içerisinden birim çağrılmaz; birimler arasında da geçiş yapılabilir.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum13/Ornek4)

Birimler

uMutfak

Ana Kod | uMutfak

```
var
  Form2:TCLForm;
  LblMutfak : TclLabel;
  Button2 : TclButton;

void AnaKodaGit;
{
  Form2.clHide;
}
{
  Form2 = TCLForm.Create(Self);
  LblMutfak = Form2.AddNewLabel(Form2,'LblMutfak', 'uMutfak Birimindesin!');
  LblMutfak.Align = alTop;
  Button2 = Form2.AddNewButton(Form2,'Button2', 'Ana Kod''a git!');
  Button2.Width = 150;
  Form2.AddNewEvent(Button2,tbeOnClick,'AnaKodaGit');

  Form2.Run;
}
}
```

Kod 126: uMutfak Birimi 2

Kod 128'de, birimler arasında geçiş yapıldı. Bunun için yeni bir birim oluşturup ismi "uSalon" olarak belirlendi *(bkz. Kod 127)* ve "uMutfak" birimi içerisinden "uSalon" birimine yönlendirme yapıldı *(bkz. Kod 128)*.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum13/Ornek5)

Birimler

uMutfak

uSalon

Ana Kod | uSalon

```
var
  Form3:TCLForm;
  LblSalon : TclLabel;

{
  Form3 = TCLForm.Create(Self);
  LblSalon = Form3.AddNewLabel(Form3,'LblSalon', 'uSalon Birimindesin!');
  LblSalon.Width = 200;
  Form3.Run;
}
```

Kod 127: uSalon Birimi

"uSalon" birimi oluşturulduğuna göre artık "uMutfak" sayfası içerisinde bir butona tıklandığında "uSalon" birimine yönlendirme yapılabilir.

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum13/Ornek6)

Birimler
uMutfak
uSalon

Ana Kod | uMutfak

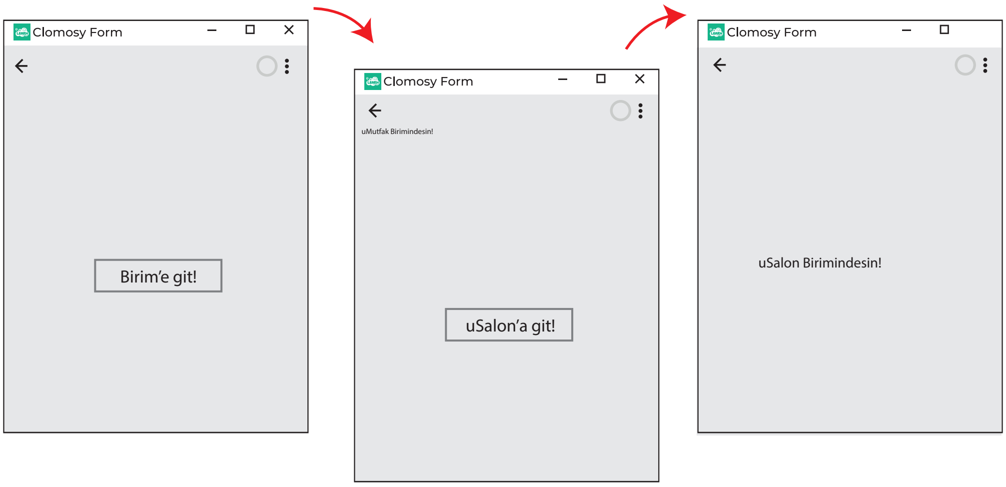
```
var
 Form2:TCLForm;
 LblMutfak : TclLabel;
 Button2 : TclButton;

void uSalonaGit;
{
 Clomosy.RunUnit('uSalon');
}
{
 Form2 = TCLForm.Create(Self);
 LblMutfak = Form2.AddNewLabel(Form2,'LblMutfak', 'uMutfak Birimindesin!');
 LblMutfak.Align = alTop;
 Button2 = Form2.AddNewButton(Form2,'Button2', 'uSalon''a git!');
 Button2.Width = 150;
 Form2.AddNewEvent(Button2,tbeOnClick,'uSalonaGit');

 Form2.Run;
}
```

Kod 128: uMutfak Birimi 3

"Ana Kod", "uMutfak" ve "uSalon" birimleri oluşturup çalıştırdıktan sonra ekrana, *Görsel 110* gibi bir çıktı gelecektir:

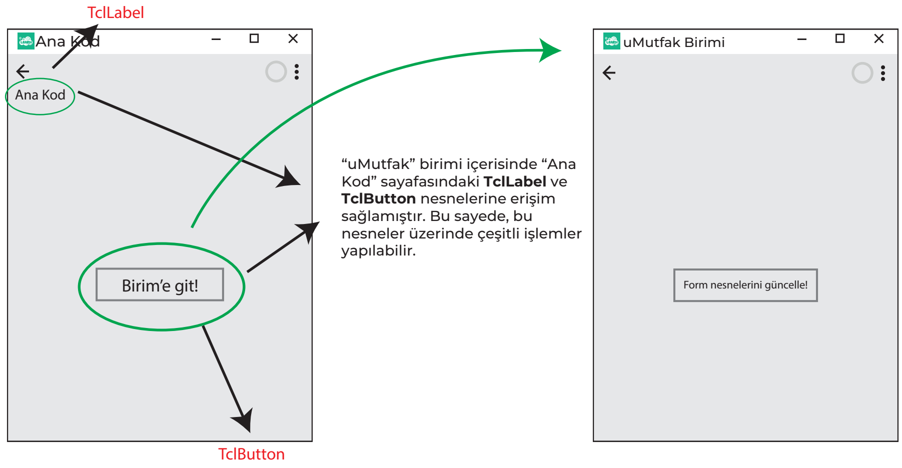


Görsel 112: Birimler Arası Geçiş Örneği

TCLUNIT

Bir önceki yöntem olan RunUnit gibi oluşturulan birimi, istenilen sayfada çağırmak için kullanılır. Bu yöntemin farkı, hangi birim üzerinden çağrıldıysa o birim içerisindeki nesnelere erişim sağlamasıdır. Bu, Clomosy'de özel bir birim çağırma yöntemidir.

Örneğin; "Ana Kod" üzerinden "uMutfak" birimine geçmek istendiğinde "Ana Kod" sayfasında bir buton ve bir etiket bulunur. *TclUnit* ile "uMutfak" birimine geçiş sağlandığında "uMutfak" biriminden "Ana Kod" sayfasındaki tüm nesnelere erişim sağlanabilir *(bkz. Görsel 111).*



Görsel 113: Birimler Arası İletişim

Kod 131’de TclUnit nasıl kullanılır kod üzerinde gösterildi. Öncelikle “Ana Kod” ve “uMutfak” birimleri oluşturuldu *(bkz. Kod 129, 130).*

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum13/Ornek7)

Ana Kod

```
var
 Form1 : TCLForm;
 BtnAnaKod : TclButton;
 LblAnaKod : TclLabel;
{
 Form1 = TCLForm.Create(Self);
 Form1.clSetCaption('Ana Kod');
 LblAnaKod = Form1.AddNewLabel(Form1,'LblAnaKod', 'Ana Kod');
 LblAnaKod.Align = alTop;
 LblAnaKod.Height = 70;
 LblAnaKod.StyledSettings = ssFamily;
 LblAnaKod.TextSettings.Font.Size = 20;
 BtnAnaKod = Form1.AddNewButton(Form1,'BtnAnaKod', 'Birim''e git!');
 BtnAnaKod.Width = 100;
 Form1.Run;
}
```

Kod 129: Ana Kod Ekranı 2

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum13/Ornek8)

Birimler

uMutfak

uSalon

Ana Kod | uMutfak

```
var
  Form2:TclForm;
  Button2 : TclButton;

{
  Form2 = TCLForm.Create(Self);
  Form2.clSetCaption('uMutfak Birimi');
  Button2 = Form2.AddNewButton(Form2,'Button2', 'Form nesnelerini güncelle!');
  Button2.Width = 250;

  Form2.Run;
}
```

Kod 130: uMutfak Birimi 4

Sırada, "Ana Kod" sayfasında oluşturulan butona tıklandığında TclUnit kullanarak "uMutfak" birimine yönlendirme yapılması var *(bkz. Kod 131).*

Bunun için başlangıçta TclUnit sınıfına ait bir nesne tanımlanmalıdır.

```
Unit1 : TclUnit;
```

Tanımlanan nesne oluşturulmalıdır.

```
Unit1 = TclUnit.Create;
```

Bu işlemler yapıldıktan sonra form üzerindeki butona tıklandığında aşağıdaki adımlar gerçekleştirilir:

- Birimler arasında yönlendirme yapılacağı için 'Unit1' nesnesinin *UnitName* özelliğine gidilecek olan birim adı yazılmalıdır.

  ```
  Unit1.UnitName = 'uMutfak';
  ```

- Gidilecek olan birimin hangi formdan çağrıldığını belirlemek için *CallerForm* özelliği ile referans belirlenmelidir. Örnekte "Ana Kod" sayfasındaki 'Form1' nesnesi verilmelidir:

  ```
  Unit1.CallerForm = Form1;
  ```

- Yapılan bu işlemlerden sonra gidilecek olan birimi başlatmak için *Run* metodu kullanılır:

  ```
  Unit1.Run;
  ```

Yukarıdaki adımlar uygulanarak "Ana Kod" sayfası güncellenmelidir *(bkz. Kod 131).*

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum13/Ornek9)

Ana Kod

```
var
  Form1:TCLForm;
  BtnAnaKod : TclButton;
  LblAnaKod : TclLabel;
  uMutfak :TclUnit;

void BirimeGit;
{
  uMutfak.UnitName = 'uMutfak';
  uMutfak.CallerForm = Form1;
  uMutfak.Run;
}
{
  Form1 = TCLForm.Create(Self);
  Form1.clSetCaption('Ana Kod');
  uMutfak = TclUnit.Create;
  LblAnaKod = Form1.AddNewLabel(Form1,'LblAnaKod', 'Ana Kod');
  LblAnaKod.Align = alTop;
  LblAnaKod.Height = 70;
  LblAnaKod.StyledSettings = ssFamily;
  LblAnaKod.TextSettings.Font.Size = 20;
  BtnAnaKod = Form1.AddNewButton(Form1,'btnAnaKod', 'Birim''e git!');
  BtnAnaKod.Width = 100;
  Form1.AddNewEvent(btnAnaKod,tbeOnClick,'BirimeGit');
  Form1.Run;
}
```

Kod 131: Ana Kod Ekranı 3

“uMutfak” birimine TclUnit ile yönlendirme yapıldığında “Ana Kod” içerisinde oluşturulan ‘LblAnaKod’ ve ‘BtnAnaKod’ nesneleri yönetilebilir.

Bir sonraki örnekte, aşağıdaki işlemler gerçekleştirildi:

- ‘LblAnaKod’ adındaki TclLabel nesnesinin metni değiştirildi.
- *clHide* metodu ile “uMutfak” birimi gizlendi.
- *CallerForm.Show* ile bir önceki referans alınan form açıldı.

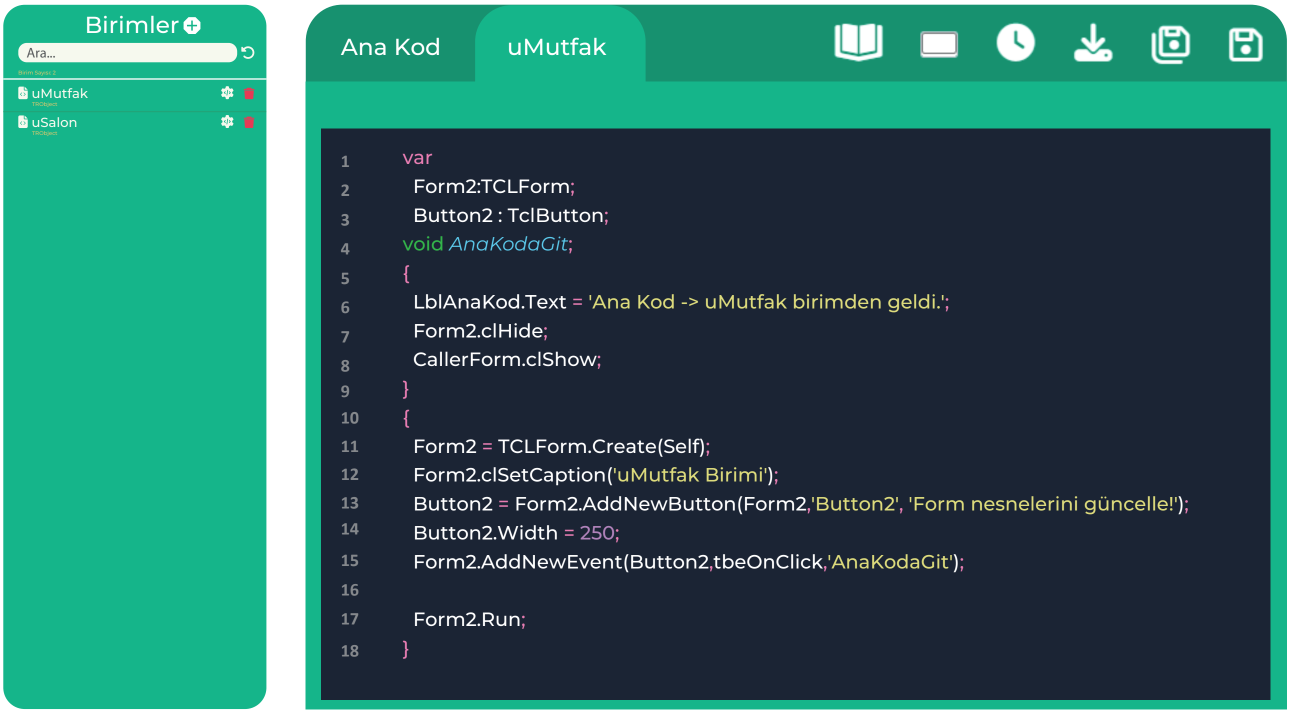
CLHIDE – CLSHOW :

- *ClHide:* Form gizleme işlemini gerçekleştirmek için kullanılır.
- *ClShow:* Form görüntüleme işlemini gerçekleştirmek için kullanılır.

Bu işlemler sonucunda, “Ana Kod” sayfasından “uMutfak” birimine geçiş yapıldı ve “uMutfak” birimindeki butona tıklayarak “Ana Kod” içerisindeki etiket nesnesinin metni değiştirildi. “uMutfak” birimi kapatıldı *(bkz. Kod 132).*

QR Okut

(https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum13/Ornek10)




```
var
  Form2:TCLForm;
  Button2 : TclButton;
void AnaKodaGit;
{
  LblAnaKod.Text = 'Ana Kod -> uMutfak birimden geldi.';
  Form2.clHide;
  CallerForm.clShow;
}
{
  Form2 = TCLForm.Create(Self);
  Form2.clSetCaption('uMutfak Birimi');
  Button2 = Form2.AddNewButton(Form2,'Button2', 'Form nesnelerini güncelle!');
  Button2.Width = 250;
  Form2.AddNewEvent(Button2,tbeOnClick,'AnaKodaGit');

  Form2.Run;
}
```

Kod 132: uMutfak Birimi 5

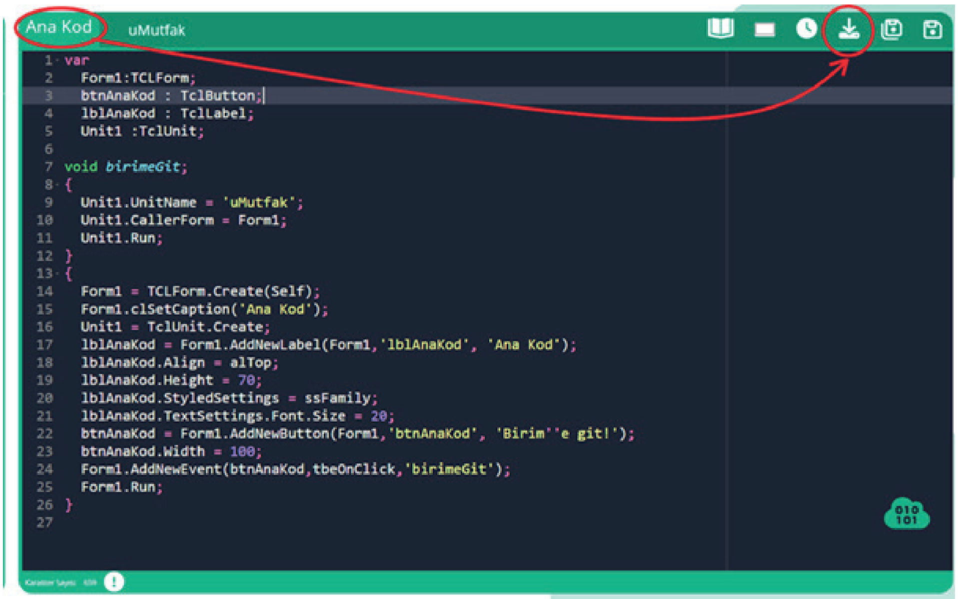
Bu şekilde referans alınan birimin nesneleri üzerinde kontrol sağlanır.

BİRİM (UNIT) İNDİRME İŞLEMİ

Clomosy platformunda web tabanlı proje geliştirirken oluşturulan birimleri gerektiğinde indirmek mümkündür. Clomosy, kullanıcılarına esneklik sağlamak amacıyla birimleri hem tek tek hem de proje olarak indirme imkânı sunar. Bu, özellikle büyük projelerde veya birçok farklı birimle çalışırken zaman kazandırıcı bir özelliktir.

Proje panelinde bulunan “Birimler” sekmesinden, birimler tek tek indirilebilir. Burada, indirmek istenilen birim seçilebilir ve indirme butonuna tıklandığında ilgili birim bilgisayara kaydedilebilir *(bkz. Görsel 112).* Bu işlem “Ana Kod” için de geçerlidir. Bu yöntem, sadece belirli birimlerle çalışmak istendiğinde veya yalnızca ihtiyaç olan birimleri indirmek istendiğinde kullanışlıdır.

BİLGİLENDİRME:
Her unit, bir dosya olarak kaydedilir ve indirilmek istendiğinde **“.tro”** uzantısı olarak indirilir. (örneğin, Unit1.tro). Bu dosyaların içinde, TRObject diliyle yazılmış kodlar bulunur.
Bunun dışında, “Ana Kod” sayfası indirildiğinde, dosya adı otomatik olarak **“MainCode.tro”** olarak adlandırılır.



Görsel 114: Ana Kod Sayfasını İndirme İşlemi

Eğer projede tüm birimler toplu bir şekilde indirilmek isteniyorsa yan menüde bulunan “IDE Ayarları” butonuna tıklanmalıdır. Ardından, “Dışarı Aktar” butonuna tıklandığında proje toplu hâlde indirilir *(bkz. Görsel 113).* Böylece projenin tüm kod dosyaları indirilebilir. Clomosy’nin sunduğu bu esnek indirme seçenekleri, projeleri yönetirken kolaylık sağlar.

BİLGİLENDİRME:
İndirilen proje, projenin adıyla kaydedilen **“.zip”** uzantılı bir dosya olarak indirilir. Bu dosya içerisinde, projede oluşturulan tüm birimler ve “Ana Kod” sayfaları bulunur.

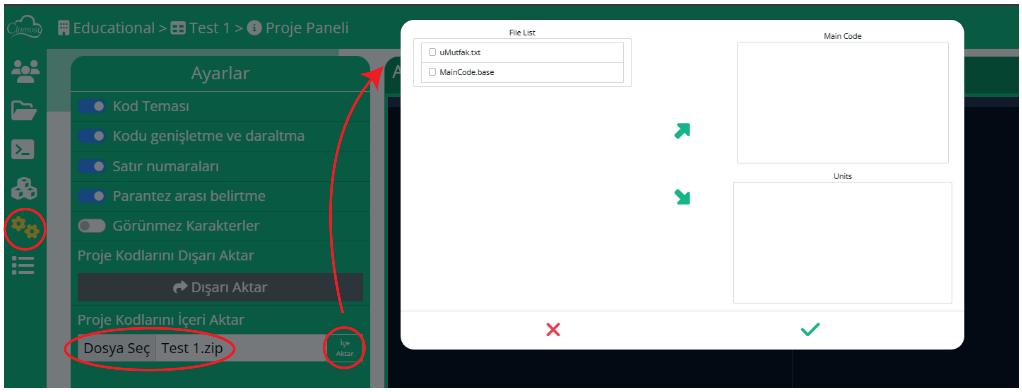


Görsel 115: Proje Dışarı Aktarma

PROJE DOSYALARINI İÇE AKTARMA

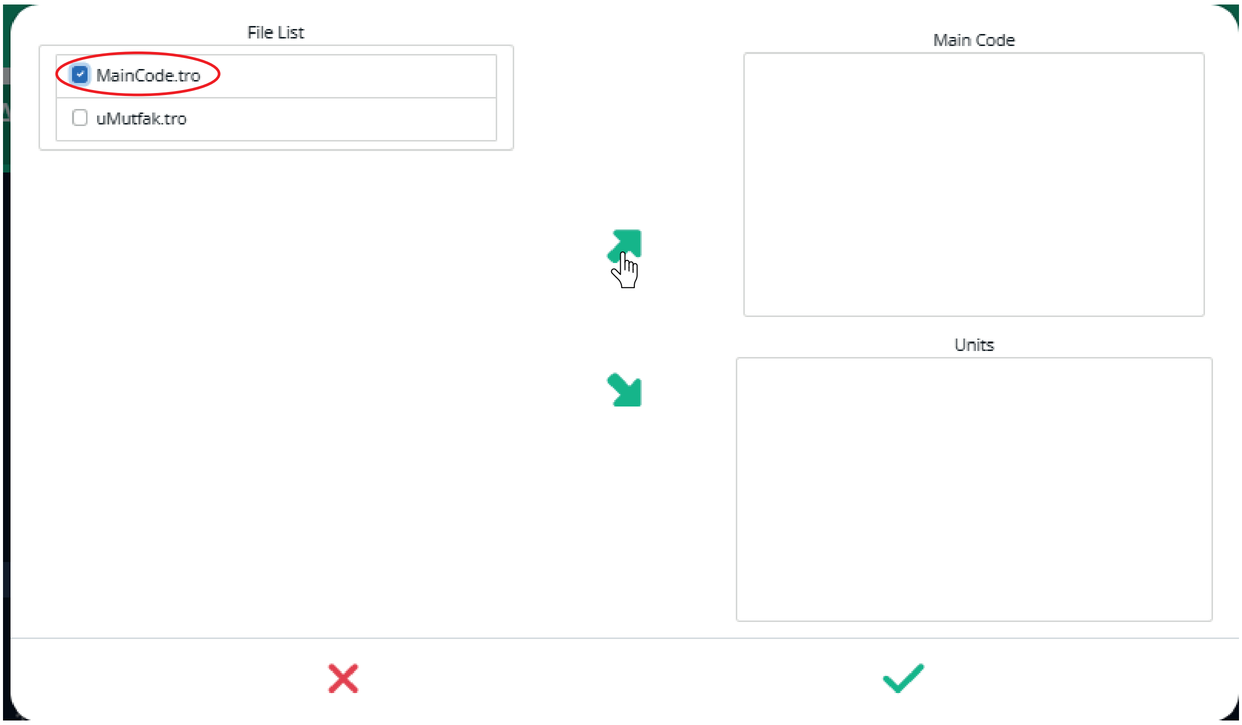
Dışarı aktarılan projeyi içe aktarma işlemi, mevcut projeleri yeniden yüklemek veya başka bir ortamda çalıştırmak için kullanılabilir. Bu işlemi gerçekleştirmek için aşağıdaki adımlar takip edilmelidir:

- **Proje İçe Aktarma Menüsüne Erişin:** Clomosy platformunda, yan menüde bulunan “IDE Ayarları” butonuna tıklayın. Bu seçenek, içe aktarma işlemini başlatmanızı sağlar.
- **Dosya Seçimi:** Açılan pencerede, “Proje Kodlarını İçeri Aktar” başlığının altında yer alan “Dosya Seç” kısmına daha önce dışarı aktarılan “.zip” uzantılı dosya belirtilir.
- **Dosyayı İçe Aktarın:** Seçilen dosyayı yüklemek için “İçe Aktar” butonuna tıklanır. Sistem, dosyayı analiz ederek içerisindeki belgeleri bir liste halinde gösterir. Bu liste, tüm birimleri ve “Ana Kod” sayfalarını içerir *(bkz. Görsel 114).*



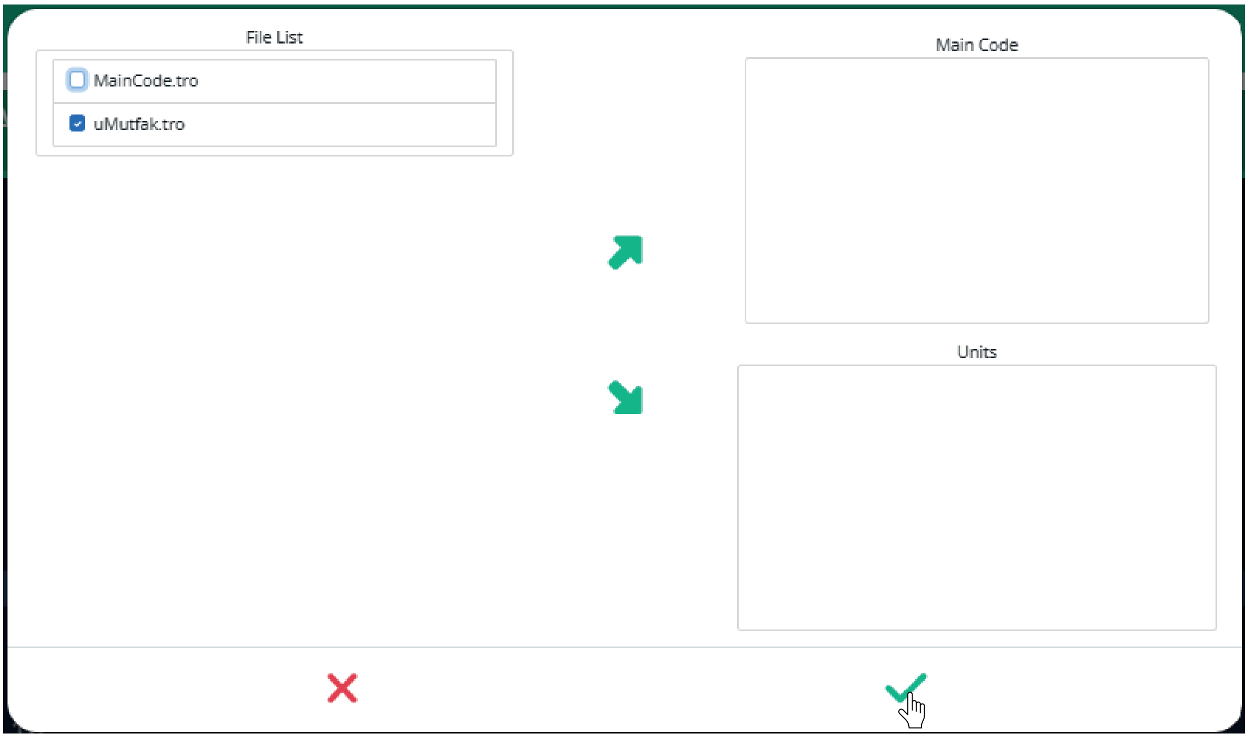
Görsel 116: Proje İçe Aktarma

Bu dosyalar arasında hangi dosyanın "Ana Kod" biriminde tutulacağı seçilmeli ve "Main Code" adındaki listeye eklenmelidir. Seçilen dosyanın yanındaki ok tuşuna basarak aktarım gerçekleştirilir. Daha sonra diğer dosyaları seçip "Units" listesine eklemek için de aynı şekilde ok tuşuna basarak aktarım yapılır *(bkz. Görsel 115).*



Görsel 117: Proje İçe Aktarma Kontrol Ekranı

Listeye belgeler eklendikten sonra sağ alt kısımdaki yeşil tik butonuna tıklayarak proje içe aktarım işlemi tamamlanır *(bkz. Görsel 116).*



Görsel 118: Proje İçe Aktarma İşlemini Tamamlama

- İçe Aktarma Tamamlandığında Kontrol Edin: İşlem tamamlandığında projenin tüm birimleri ve sayfaları platformda görünür hâle gelir. Yüklenen içerikleri kontrol ederek her şeyin doğru bir şekilde içe aktarıldığından emin olun.

Bu adımlar, projenin sorunsuz bir şekilde içe aktarılmasını sağlayarak çalışmalara kalınan yerden devam edilir.

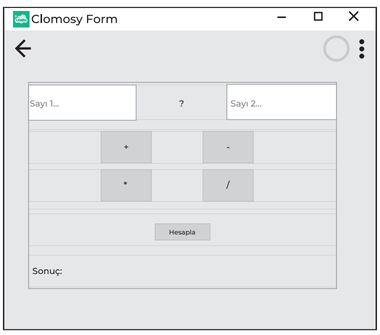
Alıştırmalar

1. Bir birimi oluşturma, düzenleme ve silme adımlarını tekrar hatırlayarak bir proje içinde birimleri nasıl yönetebileceğinizi gözden geçiriniz.

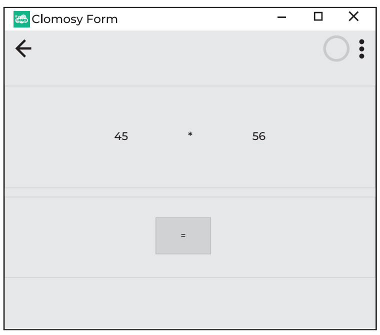
2. Farklı birimlerdeki bileşenlere erişim sağlamak için hangi yöntemlerin kullanılabileceğini örneklerle açıklayınız.

3. **Birimler Arasında Geçiş Yapma Senaryosu:**
   - uGirisEkrani: Bu birimde bir "Giriş Yap" butonu oluşturunuz. Kullanıcı bu butona tıkladığında giriş yapıldığı varsayılarak "uAnaEkran" birimine geçiş yapılacaktır.
   - uAnaEkran: Bu birimde ise kullanıcıya "Çıkış Yap" butonu sununuz. Kullanıcı bu butona tıkladığında tekrar "uGirisEkrani" birimine geçişi sağlayınız.

4. **Bir Matematik İşlemleri Yönetim Uygulaması Oluşturunuz:**
   Uygulamanın Amacı: Kullanıcıların temel matematik işlemlerini (+, -, *, /) yapabilmesini sağlayan bir uygulama tasarlanmalıdır. İşlemler bir formda yapılacak ve işlem sonuçları farklı bir birimde görüntülenecektir. Bunun için;
   - "Ana Kod" sayfasında 1. ve 2. sayıyı alabilmek için bir TclEdit, operatörü seçebilmek için 4 ayrı TclButton (+, -, *, /), "Hesapla" butonu için TclButton ve sonuç gösterebilmek için TclLabel nesneleri oluşturulacaktır *(bkz. Görsel 117)*.
   - "uHesaplama" adıyla bir birim oluşturulacaktır.
     "Ana Kod" sayfasında kullanıcıdan sayı değerleri ve seçilen operatör alınacak, "Hesapla" butonuna tıklanarak "uHesaplama" sayfasına yönlendirilecektir. Eğer veriler boşsa kullanıcı diğer sayfaya yönlendirilmemelidir *(bkz. Görsel 117)*.
   - "uHesaplama" biriminde 3 ayrı TclLabel (1. sayı, 2. sayı ve operatörü görüntülemek için) ve hesaplama yapabilmek için TclButton nesnesi oluşturulacaktır (*bkz. Görsel 118)*.
   - "uHesaplama" biriminde "=" butonuna basıldığında "Ana Kod" sayfasından gelen değerlerle (1. sayı, 2. sayı ve operatör) hesaplama yapılacak ve sonuçla beraber tekrar "Ana Kod" sayfasına yönlendirilecektir *(bkz. Görsel 118)*.



Görsel 119: 4.Soru Arayüz Tasarımı 1



Görsel 120: 4.Soru Arayüz Tasarımı 2

https://github.com/Clomosy/EgitimKitabi/tree/main/Bolum13

Ek Kaynaklar

- TclUnit Kullanımı

  https://www.docs.clomosy.com/TclUnit

  https://www.youtube.com/watch?v=TzcAwn6A3CA&t=35s